Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Dynamická detekce útoků pomocí Intrusion detection systému Suricata

Dynamic Threat Detection Using Intrusion Detection System Suricata

2015 **Bc. Pavel Pustówka**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Pavel Pustówka**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Dynamická detekce útoků pomocí Intrusion detection systému Suricata
Dynamic Threat Detection Using Intrusion Detection System Suricata

Zásady pro vypracování:

Z důvodu čím dál většího nárůstu bezpečnostních incidentů v počítačových sítích je nutno nasazovat pro jednotlivé segmenty síťových infrastruktur detekční systémy, které jsou schopny rozpoznat případné anomálie. V případě detekce hrozby musí systém reagovat v co nejkratším čase a pokud možno autonomně realizovat protiopatření, která útok eliminují či minimalizují. Cílem diplomové práce je otestovat a analyzovat chování systému Suricata v případě detekce hrozby a následně navrhnout a implementovat bezpečnostní soubor pravidel, která budou dynamicky aplikována na postiženou část sítě.

Body zadání:
1. Popis problematiky bezpečnosti počítačových sítí, popis a principy IDS systémů obecně.
2. Detailní rozbor nejčasteji se vyskytujících útoků v počítačových sítích a jejich identifikace.
3. Návrh a praktická realizace testovací síťové topologie.
4. Funkční analýza a výkonostní testování IDS Suricata při nasazení v testovací topologii.
5. Grafické zpracování výsledků testování.
6. Na základě získaných výsledků - návrh a praktická implementace skriptů pro dynamické nasazení bezpčnostních pravidel v ohrožených uzlech sítě.

Seznam doporučené odborné literatury:

[1] The Practice of Network Security Monitoring: Understanding Incident Detection and Response, Richard Bejtlich, ISBN-13: 978-1593275099, 2013

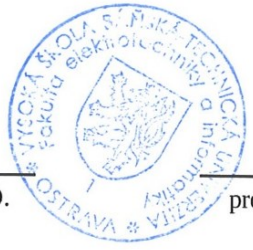
[2] Applied Network Security Monitoring: Collection, Detection, and Analysis , Chris Sanders, Jason Smith, ISBN-13: 978-0124172081, 2013

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Filip Řezáč**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015



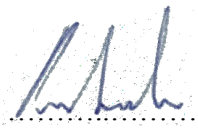
doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *4. května 2015*



.....
podpis studenta

Poděkování

Rád bych poděkoval vedoucímu této diplomové práce Ing. Filipu Řezáčovi za odbornou pomoc a konzultace. Dále bych chtěl poděkovat blízké rodině za morální a finanční podporu při studiu na vysoké škole.

Abstrakt

Tato diplomová práce poukazuje na alternativní možnosti v oblasti bezpečnosti počítačových sítí. Vedle firewallu a různých access listů v softwarové nebo hardwarové podobě, nabízí tato práce pohled na řešení problematiky bezpečnosti počítačových pomocí detekčního systému Suricata. Myšlenka uchovávání citlivých dat v elektronické podobě si nese s sebou riziko zneužití v případě využití dnešních sítí a možnosti přístupu k těmto údajům. Je to cena za mobilitu a flexibilitu a proto je nutné apelovat na vyšší míru bezpečnosti v počítačových sítí. Suricata je nástroj umožňující analyzovat provoz v sítí, oznamovat případné anomálie a působit tak jako další faktor zvyšující bezpečnost v sítích vedle firewallů, antivirů a access listů. Cílem této práce je komplexní otestování Suricaty a případná možnost využití v korporátních sítích.

Klíčová slova

Suricata; IDS; IPS; Dynamická detekce útoků; Kryptografie; Útoky; OpenVAS; Nessus; Snorby; Barnyard2; VirtualBOX; GNS3;

Abstract

This thesis shows the alternative options in the field of network security. In addition to the firewalls and access lists in software or hardware form, this work points on the issues of computer security by using detection system Suricata. The idea of storing sensitive data in electronic form carries the risk of abuse with using today's networks and their possibility of access. It is the price for mobility and flexibility, and therefore it is necessary to appeal on a higher degree of security in computer networks. Suricata is a tool that allows to analyze network traffic with possibility of reporting any threats and anomalies and it can be another factor increasing network security alongside firewalls, antivirus and access lists. The main goal of this work is a comprehensive test of this intrusion detection system Suricata with possibility of using in corporate networks.

Key words

Suricata; IDS; IPS; Dynamic Threat Detection; Cryptography; Attacks; OpenVAS; Nessus; Snorby; Barnyard2; VirtualBox; GNS3;

Seznam použitých zkratek

Zkratka	Význam
3DES	Triple data encryption standard
ARP	Address resolution protocol
CBC	Cipher block chaining
CRC	Cyclic redundancy check
DDoS	Distributed Denial of Service
DNS	Domain name system
DoS	Denial of Service
FTP	File transfer protocol
GCM	Galois counter mode
HTTP	Hypertext transport protocol
IPsec	Internet protocol security
LDAP	Lightweight Directory Access Protocol
MITM	Man in the middle
NVT	Network vulnerability test
SMTP	Simple mail transfer protocol
SNMP	Simple network management protocol
SOCKS	Socket Secure
SQL	Structured Query Language
TCP	Transmission control protocol
UDP	User datagram protocol
VoIP	Voice over Internet Protocol
VPN	Virtual private network
XML	Extensible Markup Language

Obsah

Úvod

Tato diplomová práce se snaží přiblížit otázku bezpečnosti počítačových sítí. Cílem této práce je analýza intrusivního detekčního systému Suricata.

První kapitola zasahuje do teorie počítačových sítí s úmyslem přiblížit a poukázat na obecné principy z oblasti bezpečnosti sítí. V první polovině této kapitoly jsou rozebrány různé způsoby a technologie zajišťující bezpečnost sítí minulé i současné doby. Druhá půlka se zaměřuje na principy a vlastnosti intrusivních detekčních systému.

Druhá kapitola popisuje různé útoky používané v počítačových sítích. Nachází se zde rozbor širokého spektra útoků, možnosti detekce, prevence a jejich odstraňováni.

Obsah třetí kapitoly je úvodem do praktické části této práce. Je zde navrhnutá testovací topologie a jako stěžejním bodem této kapitoly je popis instalace detekčního systému Suricata a souvisejících programů jako jsou Snorby a Barnyard2. Návrh topologie je zrealizován v simulačním programu GNS3 a virtuální stroje vytvořené za pomoci nástroje VirtualBOX byly do této topologie následně naimportovány. Součástí této kapitoly je podrobný popis konfiguračního souboru suricata.yaml, který patří mezi nejdůležitější části systému Suricata.

Čtvrtá kapitola nahlíží do jádra praktické části. Popisuje samotné testování z pohledu obou účastnících se stran. Na straně testující ukazuje vytvoření testovacího scénáře a zprovoznění testovacího frameworku OpenVAS. Obecně popisuje architekturu tohoto penetračního frameworku. Nachází se zde také testování pomocí penetračního nástroje Nessus. Na straně testované poukazuje na nainstalované služby. Kapitola také obsahuje výčet všech jednotlivých penetračních testů. Poslední podkapitolou je analýza VRT pravidel používané konkurenčním detekčním systémem Snort.

V páté kapitole jsou graficky znázorněny a popsány výsledky testování z kapitoly čtvrté. Jsou zde přehledně vypsány všechny penetrační testy v tabulce, součástí které jsou informace o úspěšnosti detekce. Dále jsou zde znázorněny možnosti grafického monitoringu Snorby a ukázky z tohoto webového prostředí. Kapitola také obsahuje graf počtu signatur a jejich rozdělení podle závažnosti detekovaných při penetračních testech.

V poslední šesté kapitole se poukazuje na další možnosti detekčního systému Suricata. Jedná se především o možnost využít Suricatu jako aktivní prvek v síti s cílem dynamicky kontrolovat provoz a zabraňovat případných hrozbám. Jsou zde popsány všechny nutné kroky pro korektní nastavení v režimu IPS, včetně úpravy topologie. Při tomto zapojení je nutná kooperace s iptables, jejíž moduly jsou využívané suricatou. Takto navržené řešení bylo testováno stejným penetračním scénářem jako v kapitole čtvrté. Výsledky testování v režimu IPS jsou popsány a znázorněny graficky. Součástí této kapitoly je i skript pro automatickou konfiguraci v režimu IPS.

1 Studijní část: Problematika bezpečnosti počítačových sítí, popis a principy IDS systémů obecně.

Na bezpečnost v počítačových sítí nebyl v minulosti kladen takový důraz, jako je dnes. V dnešní informační době, kdy jsou osobní informace často součástí nějaké databáze v internetu, je nutné nastavit určitá pravidla, která budou tyto údaje poskytovat pouze a jen pověřeným osobám.

V roce 1883 formuloval holandský lingvista axiom, který říká, že bezpečnost šifrovacího systému nezávisí na utajení šifrovacího algoritmu, ale závisí na utajení klíče. Předpokládá se, že algoritmus nepřítel zná do detailu, avšak nezná klíč [1]. Na tomto axiomu se staví myšlenka využití kryptografických algoritmů i v dnešní době.

1.1 Kryptografie

Je věda zabývající se bezpečným přenosem informace mezi dvěma stranami. Bezpečnostní specialisté spoléhají na kryptografické algoritmy a šifry zabezpečující komunikaci, která by jinak byla náchylná k odposlechům nebo jiným útokům. V podstatě je nutné přístupy k informacím autentizovat, zaručit jim integritu, důvěrnost a nepopíratelnost. Tyto čtyři faktory jsou cílem kryptografie.

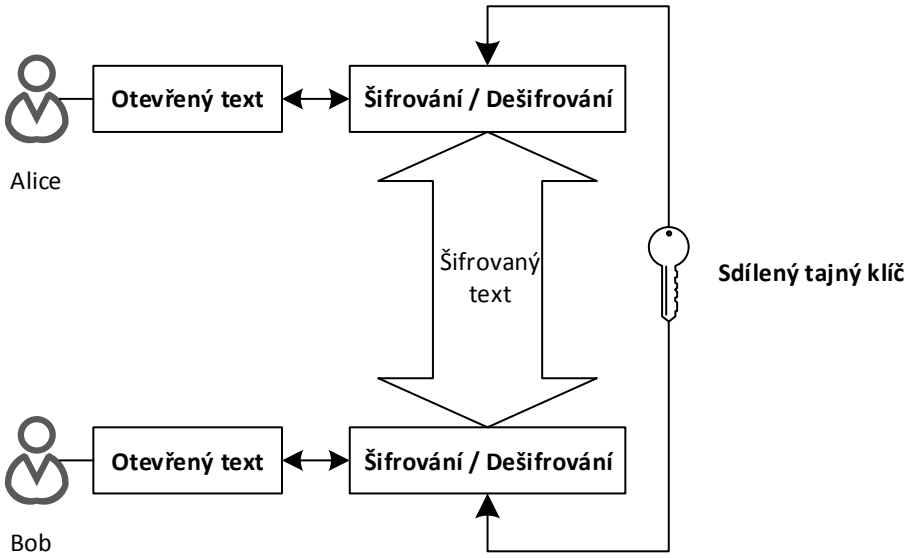
- Autentizace - je ověření identity, to znamená, že při komunikaci s druhou stranou je to právě ta entita, za kterou se vydává. Ověřování probíhá pomocí hesla. [1, 2]
- Důvěrnost - je zajištění, že data nemohou být přečtena nepovolovanými příjemci. Tuto problematiku řeší šifrování. Šifrování je proces zapouzdření dat, který transformuje data do podoby nesrozumitelného tvaru za použití různých šifrovacích algoritmů. Přečtení šifrovaných dat může pouze strana vlastnící dešifrovací klíč. [1, 2]
- Integrita - je zachování celistvosti dat při přenosu komunikačním kanálem. Narušení integrity znamená neoprávněnou modifikaci a tudíž neplatnost dat. [1, 2]
- Nepopíratelnost - je mechanizmus zajištující, že zdroj dat nemůže popřít, že informace poslal [1, 2]

1.1.1 Symetrická kryptografie

Hlavním principem symetrického šifrování, je znalost předem vytvořeného klíče oběma stranami, které budou tento klíč používat k zašifrování a dešifrování zpráv posílající si mezi sebou komunikačním kanálem. Pro tento typ klíče se používá označení sdílený tajný klíč, Obrázek 1.1. [1] Tento algoritmus se nesmí používat pro výměnu klíčů při zahájení přenosu, neboť při jeho odchycení útočníkem může dojít ke zneužití. Z toho vyplývá, že je nutné zajistit bezpečnou výměnu klíčů mezi oběma stranami. Toto řeší asymetrické šifrování. Mezi hlavní výhody symetrického šifrování patří rychlost šifrování při předávání zpráv.

Typy symetrických algoritmů

- Data encryption Standard
- Advanced Encryption Standard
- Blowfish
- Twofish
- RC4



Obrázek 1.1: Princip symetrického šifrování

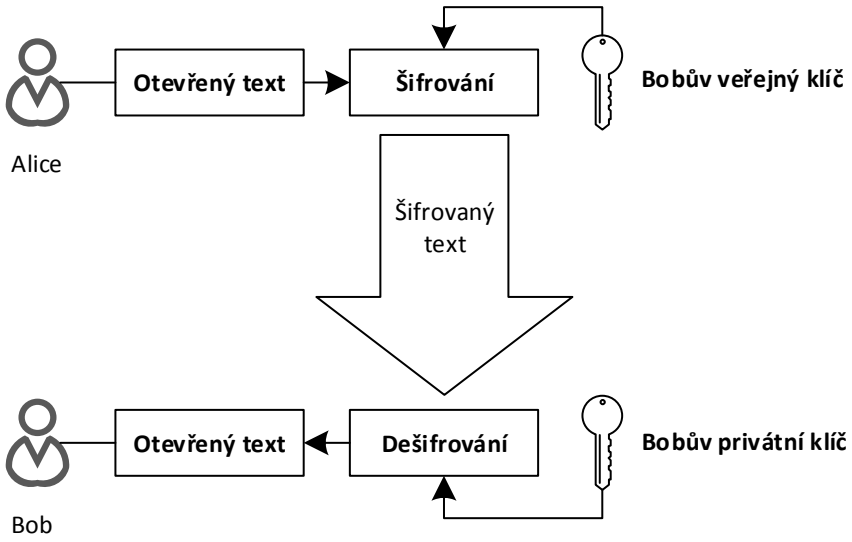
1.1.2 **Asymetrická kryptografie**

Princip šifrování asymetrickou technikou je založen na matematických vazbách mezi klíči. Každá strana si vygeneruje pár klíčů - veřejný a privátní klíč. Veřejný klíč je volně distribuován příjemcům a privátní klíč je jako tajný klíč znám pouze straně, která ho vytvořila, Obrázek 1.2.

Princip asymetricky šifrované komunikace je následující. Pokud chce Alice zaslat zprávu Bobovi, zašifruje tuto informaci bobovým veřejným klíčem, šifrovaný text komunikačním kanálem putuje k Bobovi, který tuto zprávu rozšifruje pouze svým privátním klíčem. Vztah mezi Bobovým privátním a veřejným klíčem je založen na matematické vazbě a tudíž zašifrovanou zprávu dešifruje. Výhodou tohoto typu šifrování je jeho důvěryhodnost. Používá se pro bezpečnou výměnu klíčů. Jako nevýhodou je vyšší náročnost na výpočetní výkon.

Typy asymetrických algoritmů

- Rivest, Shamir, Adelman (RSA)
- Pretty Good Privacy (PGP)
- Elliptic curve cryptography (ECC)



Obrázek 1.2: Princip asymetrického šifrování

1.1.3 **Hashovací funkce**

V situacích, kdy je nutné data zkontrolovat na případné podvržení, slouží jednocestná hashovací funkce. Tuto funkci bychom mohli definovat jako matematický algoritmus sloužící pro výpočet z vstupních dat proměnné délky řetězec pevné délky reprezentující jejich otisk. [1, 2] Nejen pro podvržení, ale také v případě porovnávání dvou databází, se kontrolují jejich hashe a v případě odlišných výpočtů se vyvolá určitá akce. Pro hashovací funkci platí tři parametry.

- Nepatrná změna vstupních dat vyvolá naprosto jiný řetězec jako výsledek hashovací funkce.
- Pravděpodobnost, kdy budou mít dvě různá vstupní data stejný otisk je téměř nulová.
- Je výpočetně nemožné, získat vstupní data zpětně z jejich otisku. Z tohoto kritéria vyplývá jednosměrnost této funkce.

Typy hashovacích funkcí

- Message digest 5 (MD5) - kryptografická funkce vytvořena Ronaldem Rivestem poskytující 128 bitový výstup. Velice rozšířená, dnes však už kryptograficky nedostačující. [2]
- RACE Integity Primitives Evaluation Message Digest (RIPEMD) - Jednosměrná funkce poskytující 128, 160, 256 a 320 bitový výstup [2]
- Secure Hash Algorithm v1 (SHA-1) - Jednosměrná funkce navržena americkou NSA poskytující 160 bitový výstup. Dnes už také kryptograficky slabá. [2]
- Secure Hash Algorithm v2 (SHA-2) - Publikována v roce 1995, dnes považována stále za bezpečnou a poskytuje otisky o bitové velikosti 224, 256, 384 a 512. [2]
- Secure Hash Algorithm v3 (SHA-3) - Vytvořena v dubnu 2013, dnes považována za standart [2]

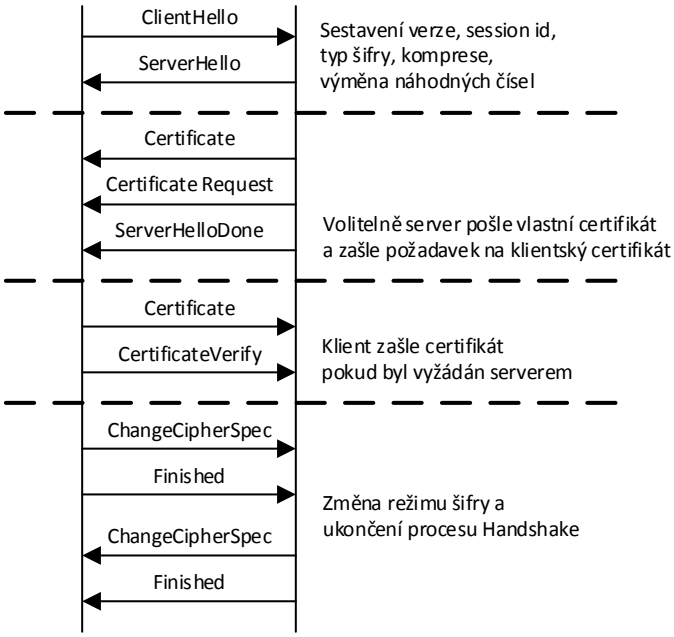
- Hash-based message authentication code (HMAC) - Technologie kombinující hashovací funkci a symetrického klíče poskytující příjemci důvěryhodnost zprávy [2]

1.2 Bezpečnostní algoritmy využívané v IP

1.2.1 SSL

Protokol zajišťující bezpečnou komunikaci mezi dvěma stranami. Vytvořen společností Netscape v roce 1994. V ISO/OSI modelu patří do relační vrstvy. Nyní je ve verzi 3.0. Z protokolu SSL 2.0 a 3.0 ještě vznikly odnože společností Microsoft – PCT resp. STLP, ve kterých zavádí některá drobná vylepšení. [1]

Protokol v podstatě zajišťuje obálku bezpečného přenosu informací z hlediska důvěrnosti, autentizace i integrity. Využívá k autentizaci možnosti asymetrické kryptografie, k důvěrnosti komunikace zase symetrické kryptografie podle vyjednaného nastavení. V úvodní fázi při sestavení spojení se používá Handshake proces uvedený na obrázku 1.3. [1]



Obrázek 1.3: Zjednodušený SSL Handshake proces

1.2.2 TLS

Tento protokol je evolučním vývojem protokolu SSL. Za vytvořením tohoto protokolu stojí skupina IETF, která v roce 1997 zdokonalila původní třetí verzi protokolu SSL a vytvořila nový standart TLS, kde byly implementovány důležité funkce. [1]

TLS oproti SSL zjednodušuje výpočet hashovací funkce ve zprávě CertificateVerify, TLS počítá otisk pouze ze zprávy nikoliv hlavičky. Dále je zavedení HMAC autentizace, vynucení 3DES namísto původní volitelné možnosti. [1]

Verze TLS 1.1 z roku 2006 již nevyžaduje restartování sezení v případě nevyžádaného pádu spojení. Oproti dřívější verzi se také zlepšila důvěrnost ve smyslu zákazu užití exportních algoritmů, změnila se práce s inicializačními vektory jako ochrana proti CBC útokům. [1]

Nejnovější verze 1.2 z roku 2008 řeší další detaily. Změny se týkají hashovacích funkcí, SHA-256 v základní variantě a použití GCM pro šifrování a autentizaci. [1]

1.2.3 IPSec

Je architektura zajišťující autentizaci, integritu, a šifrování v protokolu IP, řešenou pomocí dynamicky navazovaných zabezpečených tunelů na síťové vrstvě OSI RM. Není závislá na konkrétních algoritmech, na koncích tunelů se šifrovací a autentizační algoritmy dohadují. Ochranu proti odposlechu řeší (Security Association, SA), která spravuje bezpečnostní algoritmy a klíče v tunelech. SA se periodický mění a je určeno různě pro jednotlivý směr spojení. [3]

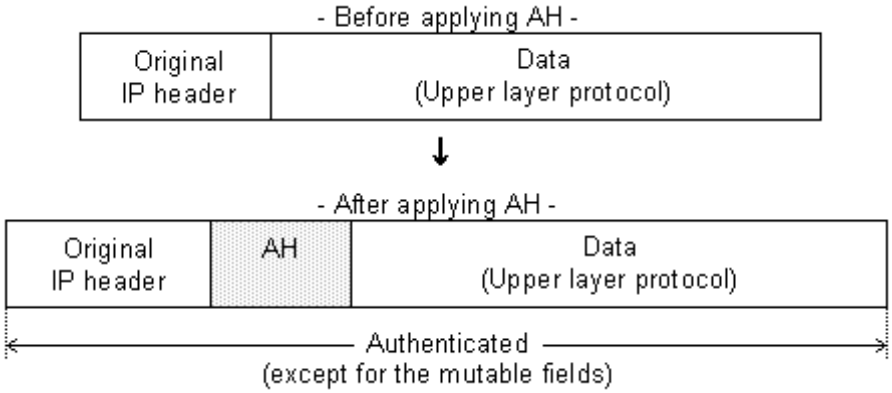
O dohodu mezi konci tunelů se stará protokol IKE (Internet Key Exchange) což je konkrétní implementace protokolu pro bezpečnou dohodu klíčů vyhovující obecnému rámci pro protokoly tohoto určení nazývanému ISAKMP (Internet Security Association and Key Management Protocol). [3]

IPSec může pracovat ve dvou režimech, transportní a tunelovací. Transportní režim (využíván nejčastěji pro spojení 2 klientů) využívá stávající IP hlavičku, kde je IPSec jako rozšíření, na druhou stranu tunelovací režim (tento režim lze využít pro VPN) využívá nového IP paketu na zabalení hlavičky IPSec. Data jsou v prvním případě šifrovaná samostatně, hlavičky jsou otevřené a ve druhém případě je šifrován celý zabalený IP datagram. [1]

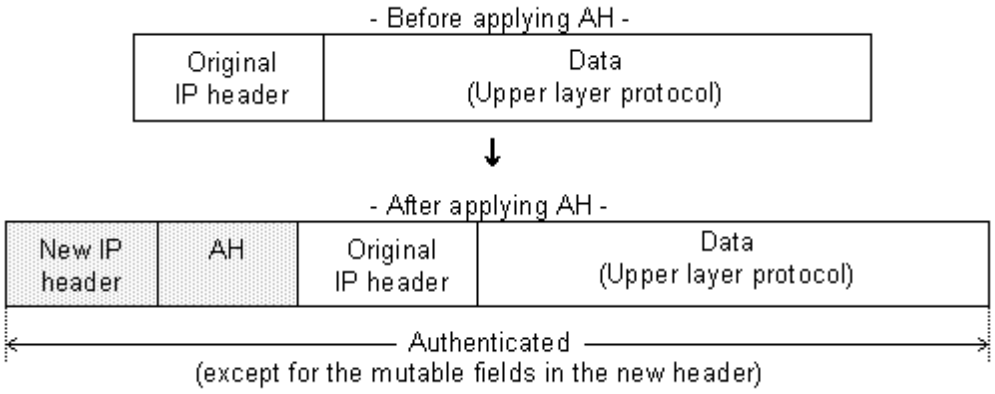
Parametry pro autentizaci a šifrování jednotlivých paketů se přenášejí přímo v těchto paketech v pomocných hlavičkách Autentication Header a Encapsulated security payload: [3]

Autentication Header

AH zabezpečuje autentizaci odesílatele a chrání integritu IP hlavičky i přenášených dat a také obsahuje ochranu před útoky zachycením provozu a jeho zopakovaným přeposíláním (tzv. replay útoky). [3] Obrázek 1.4 a Obrázek 1.5 znázorňují možné operační režimy.



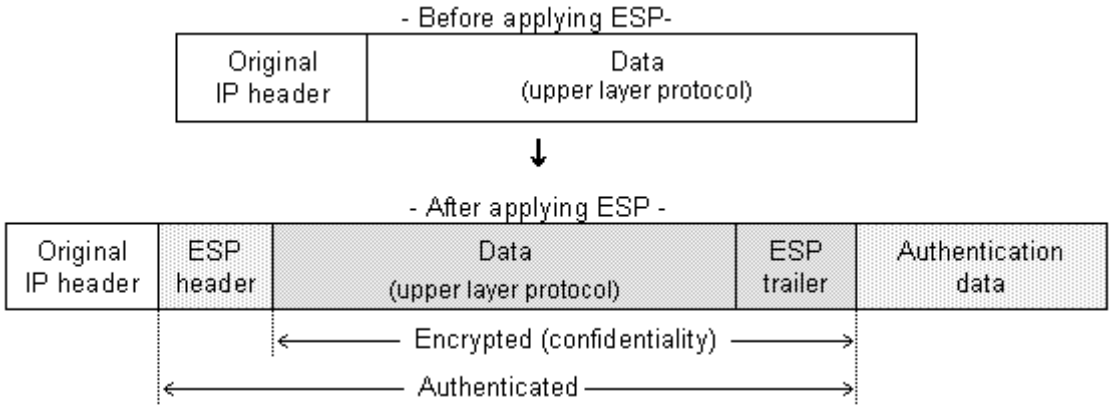
Obrázek 1.4: AH v transportním módu [4]



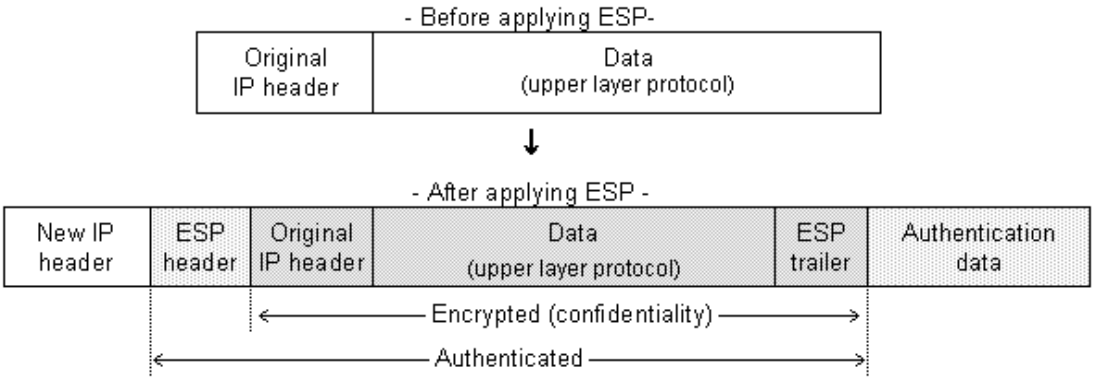
Obrázek 1.5: AH v tunelovacím módu [4]

Encapsulated security payload

ESP hlavička může zajistit všechny funkce Autentizační hlavičky a navíc přidává šifrování. V transportním režimu uložena hned za IP hlavičku nebo v tunelovacím režimu před zapouzdřený IP datagram. Za ESP hlavičkou následuje šifrovaný blok dat. ESP v transportním a tunelovacím režimu znázorňuje Obrázek 1.6 a Obrázek 1.7.



Obrázek 1.6: ESP v transportním módu



Obrázek 1.7: ESP v tunelovacím módu

1.2.4 **SSH**

SSH (Secure Shell) je v informatice označení pro program a zároveň pro zabezpečený komunikační protokol v počítačových sítích, které používají TCP/IP. SSH byl navržen jako náhrada za telnet a další nezabezpečené protokoly určené pro vzdálenou správu, převážně ty, které posílají heslo v nezabezpečené formě a umožňují tak jeho odposlechnutí při přenosu v počítačové síti. Šifrování přenášených dat, které SSH poskytuje, slouží k zabezpečení dat při přenosu přes nedůvěryhodnou síť, jako je například Internet. [1]

SSH zabezpečuje autentizaci obou účastníků komunikace, transparentní šifrování přenášených dat, zajištění jejich integrity a volitelnou bezeztrátovou kompresi. Server standardně naslouchá na portu TCP/22. Označení „Secure Shell“ je mírně zavádějící, protože nejde ve skutečnosti o náhradu shellu ve smyslu interpret příkazů. Název byl odvozen z existujícího programu rsh, který má podobné funkce, ale není zabezpečený. [1]

SSH verze 2 oproti původní nabízí navíc Diffie-Helmanův algoritmus pro výměnu klíčů, kontrolu integrity pomocí MAC funkce a řízení libovolného počtu shellů pomocí jednoho SSH spojení. Protokol má velmi dobře navrženou vnitřní architekturu na oddělené vrstvy. Je flexibilní. Funkce na transportní vrstvě srovnatelná s TLS. Vrstva autentizace uživatele je navržena pro snadné rozšíření vlastními autentizačními metodami. Vrstva spojení nabízí použití více podružných relací přenášených jedním SSH spojením, které je srovnatelné s BEEP (Block Extensible Exchange Protocol) což není například vlastností TLS. [1] SSH je používáno jako bezpečná náhrada starších protokolů a nabízí i nové vlastnosti:

- Náhrada protokolu Telnet, práce na vzdáleném počítači přes nezabezpečenou síť,
- náhrada protokolu Rlogin, přihlášení na vzdálený počítač,
- náhrada protokolu Rsh, spouštění příkazů na vzdáleném počítači,
- tunelování spojení,
- přesměrování TCP portů a X11 spojení zabezpečeným kanálem,
- bezpečný přenos souborů pomocí SFTP nebo SCP,
- automatické vzdálené monitorování a management serverů,
- bezpečné připojování složek na vzdáleném serveru jako souborový systém na lokálním počítači použitím SSHFS,
- prohlížení webu přes šifrované proxy spojení s SSH klientem, který podporuje SOCKS protokol,
- plnohodnotnou šifrovanou VPN (pouze OpenSSH server a klient, kteří tuto vlastnost podporují).

1.3 **Firewally**

Firewaly jsou softwarové nebo hardwarové řešení sloužící pro oddělení důvěryhodné a nedůvěryhodné sítě. Často jsou nasazovány k oddělení sítě nebo cílové stanice. Podle předem nastavených pravidel provádí firewall filtraci síťového provozu ven a dovnitř sítě. Může tyto data

propustit, zahodit nebo ignorovat. Správně nastavený firewall zahodí vše, čemu nerozumí, nebo co není explicitně dovoleno. Můžeme je rozdělit do tři kategorií.

- Paketové filtry - tento typ firewallu filtruje data nahlížením do hlaviček paketů a porovnává jej s pravidly. Pakety, které se neshodují, zahodí. [2] Nerozumí vyšším vrstvám, pracují s IP adresami na třetí vrstvě. Jsou rychlé a nenáročné na systémové zdroje. [5]
- Stavové firewally - používají stavovou inspekci pro filtrování. Ta porovnává komunikaci probíhající a novou. V prvním případě povolí tok dat a v druhém případě porovná spojení podle pravidel, a pokud se nezjistí shoda, data se zahodí, v opačném případě se vytvoří nový záznam v tabulce stavů. Jsou preferovány před paketovými filtry, neboť dokáží snížit zátěž a zvýšit propustnost tím, že testují jen nové spojení. Nesledují obsah zpráv. [2]
- Aplikační firewally - třetí generace firewallů umožnují navíc kontrolu spojení na vyšších vrstvách OSI modelu. Analyzují obsahy zpráv komunikací využívající protokolů aplikačních vrstev jako je HTTP, FTP, P2P a porovnávají je s pravidly. Jelikož aplikační protokoly mohou být v systémech špatně rozpoznatelné v případě změn v operačním systému nebo z důvodu nekorektního značení dat určité aplikace na výstupu, je nutné vynaložit důkladnější kontrolu těchto zpráv a tím se zvyšují nároky na zdroje. [2]

1.4 VPN

Virtuální privátní sítě (VPN) jsou prostředkem k vytváření vlastních soukromých sítí napříč internetem. Při tom je však zachováno zabezpečení a flexibility jako při použití vlastní infrastruktury. Principem VPN je tunelování šifrovaných dat přes sdílenou síť za použití autentizace mezi jednotlivými konci tunelů. Tunelem zde rozumíme virtuální dvoubodové spojení přes sdílenou infrastrukturu, které nese pakety jednoho protokolu zabalené v jiném (nebo i tomtéž) protokolu. Výhodou VPN je nízká cena a flexibilita. [3]

V praxi se pak klient přihlašuje s lokálního počítače za pomocí softwarového agenta na cílový VPN koncentrátor, který má funkci vstupní brány do cílové sítě. [3]

Z praktického hlediska pak implementace je možná na třetí a čtvrté vrstvě OSI modelu. Je to z následujících důvodů. Implementace na druhé vrstvě by vyžadovala rozšifrování a dešifrování na každém směrovači a tudíž je to neefektivní. Na transportní vrstvě je to použitelné pouze pro spojový protokol TCP a na sedmé vrstvě by musely být aplikace implementačně připravené. Z praktického hlediska je nejlepší řešit tuto problematiku na vrstvě síťové. Takovéto řešení se nazývá IPSec. [3]

1.5 Principy IDS systémů

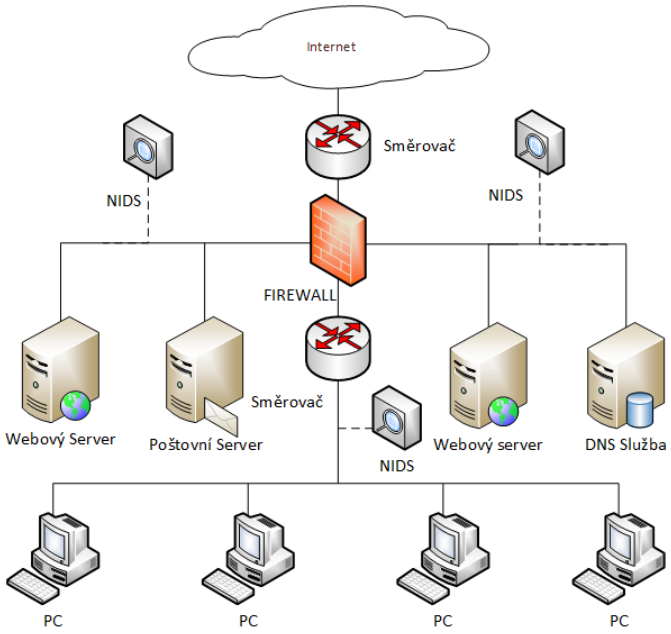
Pod zkratkou IDS z angličtiny si představme systémy detekce neoprávněného vniknutí do počítačové sítě. Vniknutí, nebo neoprávněný přístup může provést útočník z místní sítě nebo z Internetu. Správně nakonfigurované identifikátory IDS můžou útočníka odhalit, popřípadě upozornit administrátora o útoku zápisem do logu. Tyto systémy mohou upozorňovat i v případě, kdy útočník nenapadá přímo sít, ale odposlouchává ji.

IDS systémy mohou být hardwarové nebo softwarové. Hardwarové jsou dražší, nicméně dostává se nám do rukou výrobcem přednastavené zařízení, které stačí zapojit. Každý výrobce dává do svých IDS systému svoje vlastní jádro, které obsahuje signatury. Tyto systémy si dokážou stahovat aktualizace jádra a přidávají si do svých databází další nové signatury. Naopak softwarové IDS bývají levnější, některá open-source vytvářená komunitou jsou dokonce zadarmo, avšak jsou kladeny větší nároky na administrátora. Nebývají přednastavené jako hardwarové a je jen na administrátorovi jaké pravidla si vytvoří, aby ochránil svou sít před napadením. IDS řadíme do tří kategorií. [6]

- Network intrusion detection system (NIDS)
- Host intrusion detection system (HIDS)
- Distributed intrusion detection system (DIDS)

1.5.1 Network intrusion detection system (NIDS)

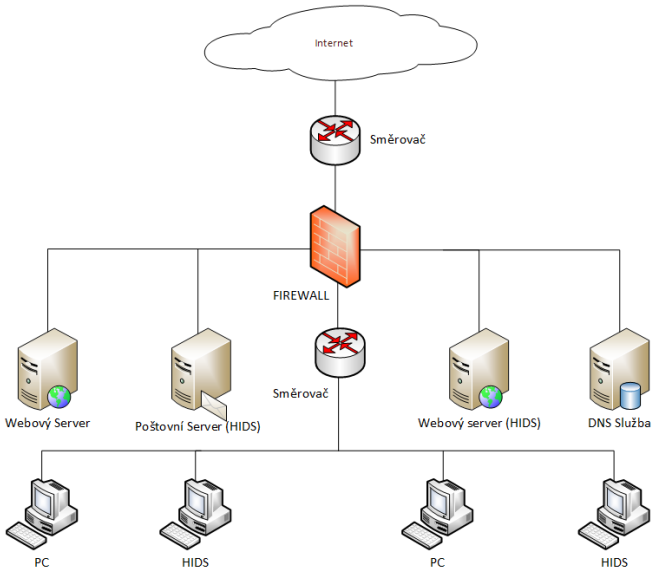
Network, neboli síťové IDS se nasazují na strategické segmenty v sítí, kde se kontrolují síťové pakety. Je nutné ale, aby síťová karta NIDS byla zapnutá v naslouchacím modu, což není její výchozí nastavení. Tento mód umožňuje analýzu veškeré komunikace na daném segmentu i v případě, že nejsou pakety explicitně směřovány. To vytváří potenciálně bezpečnostní riziko vysoké míry falešných poplachu zkontrolovaných dat. [6]



Obrázek 1.8: Příklad NIDS sítě [6]

1.5.2 Host intrusion detection system (HIDS)

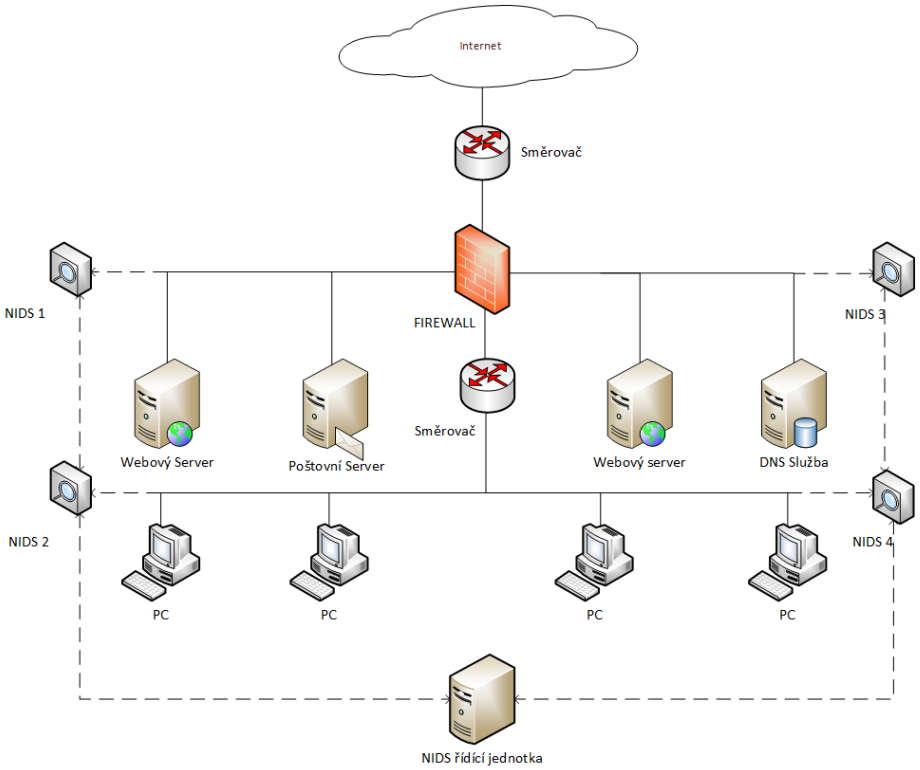
HIDS se oproti NIDS liší ve dvou základních parametrech. HIDS nemá zapnutý naslouchací mód a poskytuje monitoring pouze tomu hostu, před kterého je připojen. Lze tak nastavit jednotlivá pravidla na cílové stanice. Ne všechny typy síťových karet mají možnost přepnout se do naslouchacího režimu, proto je vhodnější v některých případech tato varianta. Lze tak například chránit e-mailový server proti vnitřním chybám, které mohou být zneužity útočníky. Výhodou je odhalení útoku přes zašifrovaný kanál, slabinou však napadení hostitelského systému. [6]



Obrázek 1.9: Příklad HIDS sítě [6]

1.5.3 Distributed intrusion detection system (DIDS)

NIDS vzdáleně komunikují s centrální stanicí, kde nahrávají své logy, nebo mohou stahovat aktualizace signatur. Jednotlivá IDS v systému mohou mít různá pravidla, první IDS může pracovat v síťovém režimu, druhá v režimu host. IDS snímače mohou být v kombinaci NIDS a HIDS. Komunikace do centrálního NIDS probíhá přes lokální nebo šifrovanou VPN síť. [6]



Obrázek 1.10: Příklad DIDS sítě [6]

2 Detailní rozbor nejčastěji se vyskytujících útoků v počítačových sítích a jejich identifikace.

Je nutné si uvědomit, že v dnešní době je kyberzločin velmi populární téma ve světě. Zcizení citlivých dat může mít katastrofální následky. Vydírání, krádeže identit nebo odčerpávání financí z účtů poškozených jsou jedny z mnoha důsledků zanedbání základních bezpečnostních pravidel. Proto je nutné apelovat na zabezpečení kritických systémů a učit lidi, jak si zachovat bezpečné internetové prostředí. V následující kapitole jsou vypsány nejběžnější útoky v prostředí počítačových sítí podle různých kategorií.

2.1 Útoky na klienta

Těmito útoky se rozumí především napadení klientských počítačů. V dnešní době výkonných počítačů je nutné tyto stanice zabezpečovat. Stanice se starým antivirovým a operačním systémem jsou velice zranitelné a technicky nezkušený uživatel připojený k internetu se tak může stát obětí vůči útokům typu virů, mallwarů a dalších. Proto je nutné být obezřetný a používat prostředky, které toto riziko snižují na minimum. Zneužití počítačů se slabou ochranou, může mít fatální následky.

2.1.1 Mallware

Mallware je termín označující skupinu softwaru, jehož úmyslem je napadnutí systému pro následné zneužití třetí stranou. Dělí se do různých kategorií, podle ohrožení od nejnižších jako jsou nevyžádané reklamy po trojské koně, které s sebou nesou vysoké riziko ohrožení, například zcizení dat. Na trhu však existuje spousta produktů, která se odstraňováním mallwaru zabývá. Od jednoúčelových balíčků po komplexní antivirové ochrany zahrnující kontrolu identity uživatele a ochrany v reálném čase.

- Adware - je označení pro mallware, jehož úkolem je podstrčit uživateli reklamu, kterou si předtím nevyžádali. Může být součástí instalačního balíčku nějakého programu. Adware pak přesměrovává uživatele na cílené stránky, nebo zobrazuje reklamu při zapnutí klientské aplikace. Nenese velká rizika, avšak může uživatele značně otravovat a omezovat. [2]
- Spyware - je software shromažďující data určitého typu o uživateli. Představuje tak o mnoho vyšší hrozbu než předchozí skupina. Získané informace mohou být použity pro cílenou reklamu [2]. Může však sbírat soukromá data, jako jsou osobní hesla, čísla kreditních karet až po monitoring e-mailové schránky uživatele. Odstraňovaní spywaru je v mnoha případech komplikované a vyžaduje zásah technika.
- Viry - jsou nebezpečné počítačové programy, které se můžou replikovat. Jejich odstraňování z počítače může být problematické a vyžaduje technicky zkušenějšího uživatele. Ovšem se známými viry si většinou antivirové programy poradí bez problému, od jejich detekce až po kroky vedoucí k trvalému odstranění ze systému. [2]

- Worms - jsou softwaroví červi, jejichž úkolem je prolézt počítačovou sítí a útočit na nezáplatované systémy, které jsou zranitelné. Rozšířený červ může vyvolat nějakou akci v určitý moment a ohrozit tak velké množství klientských stanic a serverů. [2]
- Trojský kůň - úkolem takových programů je získat vzdálený přístup do počítače. Dále pak útočník může mít přístup k webové kameře, může odchytávat hesla a má tak kontrolu nad samotným počítačem. [2]
- Botnety - jsou softwarové agenty, které vykonávají automaticky nějakou úlohu. Může se jednat o rozesílání spamů a nebo provádění DDoS útoky. Bývají často spravovány z jednoho místa. Botnety se dostanou do počítačů především po úspěšném napadení stanice trojským koněm. [2]
- Rootkity - je označení mallware, který pomohou udržet přístup do systému. Bývá nahrán po napadení červem a je často i jeho obsahem. Typicky se snaží zakrýt útočníkův software. [2]
- Backdoors - jsou v podstatě vlastnosti různých programů, pomocí jejichž funkce je možné proniknout do počítače oběti. Využívá se vstupu do systému bez autentizačního procesu. Z historického hlediska se backdoory implementovaly už při vzniku programu, a to z toho důvodu, aby se programátor mohl dostat se do systému v případě poruchy. Tyto způsoby získaní přístupu backdoorem by měl znát pouze programátor. [2]
- Logic bombs - charakterizuje se tím část programového kódu, který je už implementován v jinak normálně funkčním softwaru a čeká se jen na jeho vyvolání. Spuštění může být různého typu, například na odpočet nebo při obdržení určitého paketu. Jedna ze známých logických bomb s nazývající se Michelangelo, byla určena ke zničení pevných disků naplánovaná na 6. května 1990. [2]
- Advanced Persistent Threats - je dnes jedna z nejnebezpečnějších hrozeb zahrnující v sobě komplexní útoky napadající systémy, využívající nejnovější technologie. Za APT stojí často organizované skupiny, jejichž záměry jsou přístupy do systémů po dlouhou dobu a sběr citlivých informací. [2]

2.1.2 Útoky na aplikace

Jsou útoky využívající slabin různých programů instalovaných v systémech. Zneužívá se chyb v programech.

- Získání práv - v tomto případě se snaží útočná aplikace získat vyšší práva, pro následný útok. Útočníci tuto techniku používají pro získání kontroly nad systémem. Uživateli se pak v praxi objeví okno vyžadující povolení vyšších práv. [2] Dnes se tyto metody používají nejen k útokům, ale i pro servisní účely, tweekování například u mobilních telefonů, kde pomocí vyšších práv může uživatel využít další funkce vedoucí například k ladění systému.

2.1.3 Zranitelnost aplikací

- Zero-day útoky - jsou útoky úplně nového typu, na které nejsou vydané bezpečnostní protiopatření. Tyto útoky patří mezi ty úplně nejnebezpečnější, jelikož na ně není prozatím řešení. V oblasti open-source je však velice rychlá protireakce. [2]
- Buffer overflow - je jedna z chyb softwaru, kde se zneužívá špatně sepsaného programového kódu, obsahující chybu. V praxi se pak spustí útočný skript, který vyvolá přetečení zásobníku a zároveň otevře útočníkovi okno do systému nebo daný program přestane pracovat [2]. Prakticky se jedná o zapsání velkého množství dat do fronty určité velikosti, kde množství vstupních dat je větší než velikost fronty. Správně naprogramovaná fronta tuto slabinu má ošetřenou.

2.2 Webové útoky

Web je ideální prostředí pro uskutečnění útoků různých typů. Je to proto, že mnoho rádoby programátorů nabízí své služby levně a začínající firmy si tak mohou ušetřit a získat web za rozumnou cenu. Často se v těchto případech nemyslí důkladně na zabezpečení. Ve své praxi jsem se setkal s útokem na webovou službu, který umožnoval měnit zpětně v čase záznamy, které už propadly. Po provedení tohoto útoku by byly po určitém období peníze za služby zpětně vyplaceny. Proto je nutné rozmýšlet, jaké služby chce zákazník implementovat a poskytnout mu tak řešení s dostatečnou bezpečností.

2.2.1 Cookies

Jsou soubory malé velikosti zanechané stanicí na serveru. Obsahují záznam, který po příchodu uživatele na server identifikuje a tím mu nabídne určité možnosti. Například obsah košíku, různá nastavení webové stránky atd. Jelikož data cookies obsahují množství informací, zcizení takovýchto informací třetí stranou může být nebezpečné. [2]

- HttpOnly cookies - typ cookie, který brání krádeži útokem využívající cross site scriting [2]
- Persistent - cookie, které je tak dlouho v systému dokud mu nevyprší platnost [2]
- Secure cookies - cookies zasílány pouze v HTTPS, používány pro soukromá data [2]
- Session cookies - existují pouze u dané relace, po ukončení relace se odstraňují [2]

2.2.2 Header manipulation

Útočníci tuto techniku využívají především pro přesměrování datového provozu a mohou předstírat, že se vydávají za druhou stranu. V praxi pak odchytávají data v otevřené sítí, v hlavičce IP paketu přepisují informace a poté provoz přesměrovávají. Často se používá s technikou man-in-the-middle.

2.2.3 Directory traversal

Je metoda využívající špatně zabezpečených složek na serveru. Příčinou jsou nízká oprávnění pro složky nacházející se na straně serveru. Útočník pak vyzkouší modifikovaným

příkazem, zda je povolená cesta do důležitých složek, např. do cílové cesty vloží za odkaz ../../etc/passwd, i když se jedná o složku, která není webovým adresářem. Je třeba být opatrný a při konfiguraci systému zablokovat přístupy do systému.

2.2.4 Cross-site scripting (XSS)

Je narušení webových stránek, při kterých se zneužívá chyb skriptů na serverech. Útočníci pak vkládají vlastní nebezpečné skripty na webové stránky, které spustí útok, jakmile se na ně jiný uživatel dostane. XSS spoléhá na to, že uživatelé cílovou stránku znají a mají ji za důvěryhodnou. [2]

Existují dva typy XSS. První je tzv. dočasný XSS, kdy je vložen do URL nebezpečný skript, hned poté, když na něj uživatel klikne. Druhý typ je útok, při kterém se permanentně upraví obsah stránky například za pomocí odeslání komentáře s útočným skriptem do databáze serveru. Pro provedení tohoto útoku je nutné exploitovat server. [2]

2.2.5 Obrana proti XSS

Existuje obrana proti XSS, která by se dala rozdělit do tří fází:

- Oddělení vstupních dat účastníka - úkolem programátora je ošetřit uživatelská vstupní data tak, že vyjme potenciálně nebezpečná vstupní data a oddělí je od bezpečné části vstupních dat uživatele. [2]
- Ověření vstupu - úkolem je ošetřit vstup tak, že jen a pouze předem zadané vstupy budou platné. [2]
- Ověření cookies - cookies jsou obvykle nastavovány stránkou, ale to neznamená, že by je uživatel nemohl pozměnit. Jestliže stránka neověřuje cookies proti neošetřeným vstupům, může být použit XSS. [2]

2.3 Útoky typu injection

Útoky injection, neboli vkládáním se zaměřují na modifikaci uživatelských dat vkládáním příkazů do důvěryhodné komunikace. Metoda vkládání je možná u spousty aplikací a to nejen ty desktopové ale i webové.

Pro úspěšné provedení tohoto útoku je nutné, aby útočník odposlouchával data dvou komunikujících stran. V tomto případě vystupuje útočník jako proxy, kde může ovlivňovat provoz mezi dvěma stranami. [2]

Další variantou útoku vkládáním, je modifikace databáze, kterou chce útočník pozměnit. Využívá se nedostatečného ošetření zabezpečení databáze a vhodným příkazem je možné získat z této databáze citlivá data. [2]

2.3.1 SQL Injection

Útoky tohoto typu jsou nejčastěji zaměřeny na dynamické weby, kde běží nějaká databáze a zejména na ty, které jsou nedostatečně zabezpečené. Využívá se logický dotazů, které mají získat citlivá data [2]. V případě možného SQL útoku mají dále útočníci tyto možnosti:

- Mohou získat přístup do databáze, která je sdílená více aplikacemi a dostanou se tak k citlivým informacím.
- Mohou smazat údaje z databáze z určitého rozsahu nebo i celou databázi
- Mohou vkládat jednotlivé údaje měnit tak různá data, například cenu produktů.

Z toho lze usoudit, že je nutné klást vyšší nároky na zabezpečení databází. Naštěstí je ošetření databáze jednoduchou záležitostí. Mezi preventivní kroky patří:

- Odstraňování zvláštních znaku
- Kanonizace
- Používání pouze uložených procedur
- Omezování pověření aplikací

2.3.2 LDAP a XML Injection

Oproti SQL injection je LDAP útok zaměřen na lokální sítě. Cílem takovýchto útoku je získání vyšších práv. XML pracuje podobně. Díky tomu že je XML otevřený formát, který využívá mnoho webových aplikací, narůstá také počet potenciálních útoků. [2]

2.3.3 Command Injection

Jako předchozí typy útoků i tento je podobného charakteru. Cílem útočníků jsou programy a aplikace pracující s terminálovým prostředím vyžívající jejich vlastnosti a funkce. Naštěstí ošetření uživatelova vstupu umožnuje ztížit úspěch na tento útok. Mimo jiné bývá také označován jako shell injection s cílem spustit určité příkazy.

2.4 Síťové útoky

Skoro všechny předchozí formy útoku využívají jedno společné prostředí - Internet. Je proto nutné brát ohled na bezpečnost v internetovém prostředí, kde existuje největší potenciál pro útoky do cizích sítí.

2.4.1 Spoofing

Je technika podvržení informací třetí stranou do sítě. Podvrhované informace můžou být různého typu, IP adresa, DNS záznam, MAC adresa a další.

- DNS spoofing - útočník podvrhne svůj vlastní DNS záznam ve chvíli, kdy napadená strana nahlíží do DNS záznamu. Tato technika umožnuje útočníkovi přesměrovat provoz napadené strany na určitý cíl, kde může dále poskytovat falešné informace, nebo se tvářit jako důvěryhodný server.

- Gateway spoofing - podobná technika jako v předchozím případě, avšak se zde útočník snaží vystupovat jako směrovač v rámci sítě a přesměrovávat provoz. Cílem je, že veškerý provoz v rámci jednoho segmentu sítě se přesměrovává k útočníkovi.
- MAC spoofing - v případě že se útočník naboural do bezdrátové sítě, která je zabezpečená na MAC adresy, musí získat MAC adresu místního počítače připojeného do sítě. Jakmile ji nějakým způsobem obdrží, použije ji místo té své a prokazuje se tak jako ověřený uživatel.
- Caller ID spoofing - používáno ve VoIP. Útočník se vydává za různá uživatelská čísla jiná, než je to jeho vlastní. [2]

2.4.2 Sniffing

V případě úspěšného přihlášení do cizí sítě se tato technika využívá pro odchycení dat v sítí. Sniffing lze realizovat pomocí přepnutí síťové karty do promiskuitního režimu, v některých případech je nutné doinstalovat do systému potřebné ovladače. Pro úspěšný sniffing informací počítačů v přepínané sítí je možné za použití speciálních programů. V sítí s rozbočovači není nutné používat speciální programy, pakety se posílají na všechny porty. Dnes už se tato varianta z důvodu bezpečnosti nepoužívá.

2.4.3 Man-in-the-middle

Je technika odposlouchávání informací třetí stranou při probíhající komunikaci dvou dohodnutých stran. Principem je přesměrování provozu na útočníkův počítač, kde jednoduše přeposílá data na obě komunikující strany. Ovšem pro správnou funkčnost musí ze svého počítače vytvořit směrovač a donutit původní směrovač nebo nějakou z komunikujících stanic posílat na sebe data. Tato technika se nazývá ARP poissoning. Efektivně se lze bránit pouze v případě použití šifrované SSL vrstvy.

2.4.4 Replay útoky

V případě úspěšného MITM útoku, může útočník odchycená data zneužít pro následnou replikaci. Využívá se toho při odchycení sekvence autentizačního procesu, kdy se opakuje zasílání těchto dat až do určitého počtu s následným vyústěním úspěšné autentizace útočníka.

Obrana proti těmto útokům je v tom, že se data při přenosu značkují časovým razítkem. Každé nové sestavené spojení obdrží nové časové razítko, které je generováno náhodně a má omezenou dobu platnosti. Pro správnou funkčnost musí být obě komunikující strany synchronizovány a pokusy o útoky mimo časové okno jsou zahazovány. [2]

2.4.5 DNS a ARP poisoning

Pokud počítač v sítí komunikuje na portu 80 a uživatel zadává do adresního řádku prohlížeče cílovou adresu, při DNS poisoningu mu překládá tento dotaz podvržený DNS server útočníka.

Toto se váže s ARP protokolem, neboť úspěšný DNS útok je možný v případě úspěšného ARP útoku. ARP poisoning je zneužití velmi jednoduchého protokolu ARP, který nemá v sobě implementované bezpečnostní funkce. Útočník přehltí nebo přepíše svým vlastním záznamem vyrovnávací paměti přepínače a převede si veškerý provoz na svůj počítač. V této chvíli může poskytovat falešné služby klientům v síti nebo odposlouchávat provoz na přepínané síti. Existuje však několik možností jak se bránit:

- Statické ARP záznamy - přepínače se nakonfigurují staticky a ignorují tak dotazování ARP protokolu [7]
- Software detekující ARP spoofing - software v aktivním nebo pasivním režimu [7]
- Ochrana na úrovni OS - různé OS reagují odlišně, linux například ignoruje nevyžádané odpovědi, SOLARIS přijímá požadavky až po uplynutí limitu a WINDOWS umožnuje konfiguraci několika klíčů v registrech [7]

2.4.6 **DoS a DDoS**

Cílem této skupiny útoků je omezit nebo znepřístupnit informační službu. Zejména se jedná o webové služby, které využívá mnoho uživatelů. Dosažení těchto cílů může být splněno různými cestami.

- Shození nebo vypnutí služby [2]
- Vypnutím systému, na kterém běží služba [2]
- Zahlcení služby dotazy [2]
- Vypnutí nebo zahlcení bodu na cestě mezi serverem a klienty [2]
- Fyzické zničení systému, na kterých běží služby [2]

Důvodem můžou být konkurenční boje, demonstrující skupiny hackerů upozorňující na chyby nebo problémy světového měřítka. Často se jedná o vyjádření nesouladu na neprovedené změny vyžádané určitou komunitou.

Útoky vyvolané distribuovanou technikou s cílem vyřadit velmi důležité služby jsou označovány jako DDoS. Administrátor má pak problém blokovat příchozí požadavky různých stanic, jelikož ty přicházejí ze všech stran. V tomto momentě je problém odlišit, co je požadavek a co je zahlcení. Existují preventivní protiopatření a způsoby redukující dopady těchto útoků.

- Firewaly, přepínače a směrovače mohou kontrolovat přístupy, monitorující provoz a kontrolují, zda nedochází k zahlcování sítě. [2]
- IPS systémy mohou pomoct při blokování těchto útoků [2]
- Anti DoS a DDoS systémy navrženy speciálně pro tyto útoky [2]
- Konfigurace přenosového pásma a filtrování aplikací mohou pomoci zmírnit dopady těchto útoků kategorizací provozu, umožňující i při útocích oddělit kriticky důležitý provoz [2]

2.4.7 Smurf útoky

Ne každý útok dostane svůj vlastní název jako právě SMURF útok, pojmenovaný po známém kresleném seriálu. V minulosti využíval slabin v konfiguraci sítě. Cílem tohoto útoku bylo hned po podvrhnutí IP adresy rozeslat broadcastem po sítí požadavky, na které začali odpovídat okolní počítače. Důsledkem bylo zahlcení počítače oběti v sítí. Byla to určitá forma DoS útoku pro systémy využívající malou přenosovou kapacitou sítě.

2.4.8 Xmas útoky

Speciální typ útoků označovaný jako "Paket Vánočního stromu". Hlavní vlastností tohoto neobvyklého paketu bylo nastavení všech atributů na logickou jedničku, Obrázek 2.1

Flag	Name	Binary representation
CWR	Congestion window reduced	1 0 0 0 0 0 0 0
ECE	ECN echo	0 1 0 0 0 0 0 0
URG	Urgent	0 0 1 0 0 0 0 0
ACK	Acknowledgement	0 0 0 1 0 0 0 0
PSH	Push	0 0 0 0 1 0 0 0
RST	Reset	0 0 0 0 0 1 0 0
SYN	Syn	0 0 0 0 0 0 1 0
FIN	Fin	0 0 0 0 0 0 0 1
Xmas	All flags	1 1 1 1 1 1 1 1

Obrázek 2.1: Nastavení všech flagů TCP paketu na logickou 1

Avšak paket takového typu nemohl být platný za normálních podmínek. Například nemohla být aktivní značka FIN nebo END pokud se použila značka SYN. A to není vše, i ostatní kombinace mohli kolidovat a vytvářet tak neplatný paket. [2]

Pro útočníky měli pakety XMAS dvojí využití. První variantou bylo otestování cílového sytému na jeho chování, když se takovýto paket dostane na vstup síťové karty. Výsledkem tak byla identifikace systému podle jejich chování. [2]

Druhá varianta měla jiné následky. Jelikož některé směrovače nemusely zvládat zacházení z těmito pakety, docházelo k zatěžování sítě. U jiných směrovačů se důsledkem těchto atypických paketu zaplňovaly vyrovnávací paměti a docházelo ke zvyšování procesorového času, což mohlo vyústit ve výpadek nebo selhání. [2]

2.5 Útoky na bezdrátovou síť

Bezdrátové sítě otevírají nové možnosti útoků, než to bylo u drátových. Je nutné si uvědomit, že na přepínaných sítích je primitivní bezpečnost zajištěna tak, že není možné standardně odposlouchávat data. V bezdrátových sítí se informace šíří vzduchem, a tedy potenciální útočník v dosahu vysílače může data odchytávat.

2.5.1 Rogue access point

Jsou přístupové body, které mají slabou nebo žádnou zabezpečovací ochranu. Prvním rizikem je možnost, že pokud existuje takový přístupový bod v korporátní síti, jedná se o bezpečnostní díru.

Některé systémy dokážou tuto hrozbu částečně eliminovat rušením nebo zjistit polohu útočného AP.

Druhou modifikací jsou tzv. Evil twins. Jedná se o přístupové body, jejichž cílem je předstírat, že jsou ověřené a patřící do sítě. Záměrem je odchytávat data a využívat útočné techniky jako je MITM. [2]

2.5.2 Bluetooth útoky

Prvním typem je bluejacking útok, je to technika zasílání nevyžádaných zpráv dalším zařízením. Druhým je tzv. Bluesnarfing, což je neautorizovaný přístup k datům. V tomto případě se jedná o větší hrozbu, útočník může získat veškerá data z mobilního přístroje oběti. Nejefektivnější protiopatření bývá vypnutí bluetooth rádia. [2]

Mnoho uživatelů však nechává svá bluetooth zařízení viditelná ostatním, což může představovat bezpečnostní riziko. Často nechávají i spárovací pin na výchozí hodnotě předdefinovanou výrobcem. Útočník pak otestuje příslušnou kombinaci a spáruje zařízení.

2.5.3 Packet sniffing v bezdrátových sítích

V bezdrátových sítích je snadnější odposlouchávat data než v sítích přepínaných, jelikož komunikující zařízení vysílají do prostoru informace. Proto není složité tyto data odposlouchávat. Naštěstí je většina dat šifrovaných, a tudíž jejich dešifrování při odchycení třetí stranou není možné. Pro umožnění data odposlouchávat, je nutné přepnout bezdrátový adapter do promiskuitního režimu.

Z historického hlediska, šifrování WEP v bezdrátových sítích trpělo problémem s inicializačním vektorem, který byl snadno prolomitelný a útočník se tak mohl dostat do sítě. Dnešní standard WPA2 je zatím oficiálně neprolomitelný a měl by se používat jako výchozí šifrovací algoritmus.

2.6 Sociální útoky a phishing

Všechny předchozí formy útoků měly technický charakter. Tato kapitola se zaměřuje na to, jak může útočník zneužít lidské neznalosti a jaké jsou jeho možnosti. Velký podíl na úspěšnosti těchto útoků má samozřejmě to, do jaké míry jsou uživatelé obeznámení s riziky v informačním prostředí a také jestli používají bezpečné zásady.

Existují různé situace v internetovém prostředí, kde se uživatelé chovají určitým způsobem. Neplatí to pro všechny uživatele, a proto útočníci věnují spoustu času analýzou lidského chování v tomto prostředí s cílem zlepšovat své útoky. V následujících bodech budou vypsány různé techniky, které útočníci využívají.

- Shoulder surfing - v tomto případě získává útočník přístupy tím nejjednodušším případem, nahlíží přes rameno v momentě, kdy oběť zadává heslo systému. [2]
- Tailgating - útočník získává přístup do oblastí, kde nemá přístup. Typicky, když se útočníkovi podaří přerušit odhlašovací proces, zatímco uživatel už opustil stanici. [2]
- Impersonation - útočník získá důvěru někoho, kdo mu sdělí určité informace, ty pak zneužije. [2]
- Dumpster diving - získávání dat, které mají být zničena nebo skartována. Různé politiky dnešních firem jsou nastaveny tak, aby k těmto únikům nedocházelo. [2]
- Pretexting - útočník si vymyslí situaci, ve které zahraje zaměstnance zapomínající heslo a snaží se ho získat požadavkem. [2]
- Baiting - neboli návnada, útočník může zanechat ve firmě přenosný disk, na kterém zanechá trojského koně, a nadšený zaměstnanec s cílem zjistit co je na něm uloženo, si nevědomky nainstaluje trojského koně do firemní sítě. [2]
- Quid pro quo - něco za něco, může se jednat o situaci, kdy nespokojený zaměstnanec opouštějící jistou společnost předá důvěrné firemní informace cizí osobě za nějaký úplatek. [2]

2.6.1 **Hoax**

Je zpráva se smyšleným či falešným obsahem, s cílem nalákat cílového uživatele. Tyto zprávy se obvykle šíří e-maily, messengery nebo webovým obsahem. Často se tyto zprávy také objeví na sociálních sítích.

2.6.2 **Phishing**

Je forma sociálních útoků snažící se přesvědčit oběti k poskytnutí jejich osobních nebo citlivých údajů. Příkladem může být e-mail vypadající důvěryhodně, ve kterém se snaží útočník přesvědčit oběť, aby si například změnila heslo do své banky. V této zprávě bývá i odkaz pro změnu hesla, který vede na útočníkovy stránky s cílem odchytit přístupové údaje.

Prevencí je možné vyhnout se útokům tohoto typu. Někdy však i zkušení uživatelé mohou být obětmi, jelikož útoky vypadají věrohodně. Proto obecně platí, že druhá strana by po vás neměla nikdy chtít citlivé přístupové údaje. V případě problému, například u bank, je dobré řešit osobně na pobočce. Pokud toto budeme mít na paměti, je míra rizika napadení těmito útoky minimální. Existuje několik technik phishingu, které je dobré rozlišovat.

- Spear phishing - cílový phishing, úkolem je zacílit na určitou skupinu a posbírat informace z různých komunikačních kanálů. [2]
- SPIM - zvláštní forma spamu zasílána skrz IM jednomu či více uživatelům [2]
- Vishing - voice phishing, používané především ve VoIP, kde je možnost podvrhnout účastnické číslo a vystupovat jako důvěryhodná strana s cílem poškodit druhou stranu. [2]
- Whaling - útočníci se zaměřují na cíle s vysokým potenciálem, jako jsou například ředitelé firem, a pracovníci na vyšších pozicích. [2]

- Pharming - kombinace slov phishing and farming, s cílem profitovat ze systémů napadených malwarem. [2]

2.6.3 Útoky přes e-mail

Dnes asi nejpoužívanější komunikační kanál v internetovém prostředí se přímo vybízí k realizaci různých útoků. Pokud vynecháme sociální sítě, tak předávání zpráv pomocí e-mailů je jednou z nejčastější formy komunikace na internetu.

Elektronická pošta není jen zasílání pouhých zpráv, ale umožnuje připojit i přílohy podporující mediální formáty různého typu. Útočníkům se tak nabízejí další možnosti, kterých využívají pro rozesílání spamu s nebezpečnou přílohou, což můžou být různé nebezpečné skripty, programy se škodlivým kódem, viry a jiné.

2.6.4 Spam

Možná nejrozšířenější zprávy elektronické pošty dnešní doby. Spam je nevyžádaná elektronický pošta zasílána adresátovi s cílem získat jeho pozornost. Často obsahuje reklamy, nebo odkazy na cizí stránky, výjimkou nejsou ani odkazy na nebezpečné stránky.

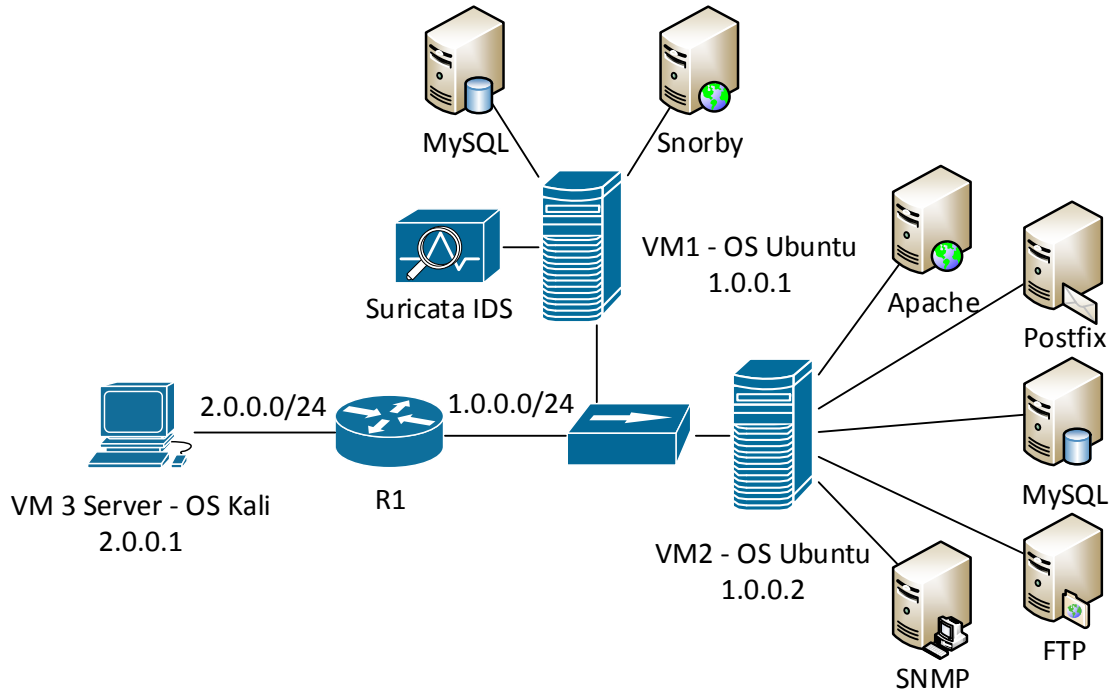
3 Návrh a praktická realizace testovací síťové topologie.

Testování probíhalo v síťovém simulátoru GNS3 ve verzi 1. 3. 0, jehož součástí byly tři servery. Tyto servery byly vytvořeny v jiném virtualizačním programu – VirtualBox. Následně byly naimportovány do prostředí GNS3, kde byly součástí testované topologie, Obrázek 3.1

Úkolem prvního serveru (VM1) bylo monitorovat síť, takže zde běžel síťový detekční systém Suricata včetně webového monitoringu Snorby. Účelem této služby bylo reprezentovat zachycené útoky graficky. Tato služba běžela pod webovým serverem Apache. Jelikož Snorby neumí přijímat výstupy přímo ze Suricaty, bylo nutné nasadit prostředníka Barnyard2. Tento program umožnuje převézt výstupy IDS z formátu unified2.alert do formátu čitelnou pro databázi, jenž využívala webová monitorovací služba Snorby.

Druhý server (VM2) reprezentoval klienta, kde běžely různé služby, jako je FTP, SNMP, MySQL, Apache atd. Všechny tyto spuštěné služby byly cílem různých útoků. Virtuální stroje 1 a 2 běžely na 64 bitovém operačním systému Ubuntu ve verzi 14.04.

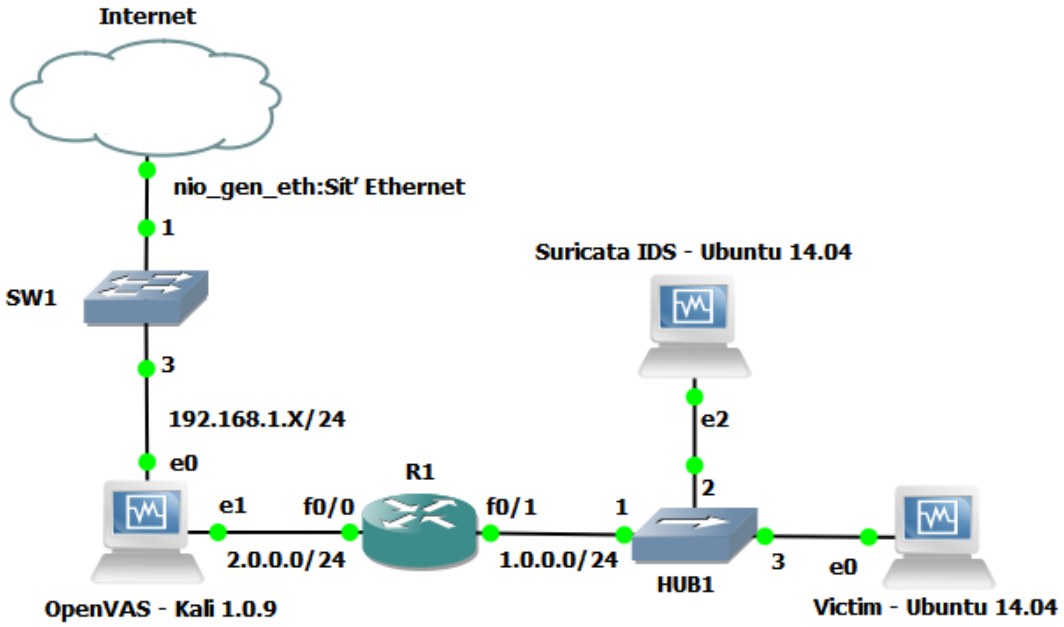
Útočný server běžel na operačním systému Kali ve verzi 1.0.9. OS Kali je bezpečností distribuce založena na Linuxu odvozena od distribuce Debian, poskytující různé penetrační testy operačního systému, webových a aplikačních služeb. Úkolem tohoto serveru bylo penetrovat klienta generováním různých forem útoků.



Obrázek 3.1: Obecná topologie pro režim IDS včetně služeb

3.1 Návrh topologie pro testování

Virtuální servery v topologii měli přidělené dvě síťové rozhraní, z nichž defaultně byla zapnutá pouze jedna – pro vnitřní sít. V případě nutného připojení na internet se aktivovala i druhá síťová karta. Na obrázku 3.2 níže vidíme testovací topologii pro režim IDS.



Obrázek 3.2: Testovací topologie v prostředí GNS3 pro režim IDS

3.2 Instalace a konfigurace IDS Suricata

Ještě před samotnou instalací IDS je třeba doinstalovat potřebné programy.

```
# sudo apt-get -y install libpcre3 libpcre3-dbg libpcre3-dev
build-essential autoconf automake libtool libpcap-dev libnet1-
dev libyaml-0-2 libyaml-dev zlib1g zlib1g-dev libcap-ng-dev
libcap-ng0 make libmagic-dev libjansson-dev libjansson4
```

Pro konfiguraci jako IPS budeme potřebovat tyto programy:

```
# sudo apt-get -y install libnetfilter-queue-dev libnetfilter-
queue1 libnfnetlink-dev libnfnetlink0i
```

Suricatu můžeme nainstalovat pomocí aptitude, nebo ji můžeme stáhnout ručně zkompilovat a nainstalovat.

3.2.1 Automatická instalace

```
# apt-get install suricata
```

V případě, že Suricata v repositářích není, přidáme ji pomocí následujících příkazů

```
# sudo add-apt-repository ppa:oisf/suricata-stable
# sudo apt-get update
```

3.2.2 Ruční instalace – doporučeno

```
# wget http://www.openinfosecfoundation.org/download/suricata-2.0.7.tar.gz
# tar -xvzf suricata-2.0.7.tar.gz
# cd suricata-2.0.7
```

Instalační balík jsme stáhli a rozbalili, v další části zkompilujeme a nainstalujeme.

```
# ./configure --enable-nfqueue --prefix=/usr --sysconfdir=/etc --localstatedir=/var
# make
# sudo make install-full
# sudo ldconfig
```

Místo přepínače –full můžeme použít –conf, pro instalaci jen s konfiguračním souborem nebo –rules pro instalaci s pravidly nebo –full kombinace těchto dvou.

Tímto se nám nainstalovala Suricata a konfigurační soubory nalezneme v adresáři /etc/suricata. Ještě před spuštěním běhu IDS je potřeba nastavit konfigurační soubor.

3.2.3 Nastavení automatického stahování bezpečnostních pravidel.

```
# apt-get install oinkmaster
```

Do souboru /etc/oinkmaster.conf file vložíme odkaz k pravidlům na webu.

```
url=http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz
# sudo oinkmaster -C /etc/oinkmaster.conf -o /etc/suricata/rules
```

Následujícím příkazem Suricatu spustíme.

```
# suricata –c /etc/suricata/suricata.yaml
```

3.2.4 Konfigurační soubor suricata.yaml

Pro nastavení režimu Suricata jako IDS/IPS/AUTO. Dále nastavíme výchozí adresář pro výstupy, logy.

```
host-mode: auto
default-log-dir: /var/log/suricata/
outputs:
  - fast:
```

```
      enabled: yes
      filename: fast.log
      append: yes // přidání záznamu do téhož souboru na konec
- eve-log:
      enabled: yes
      type: file #file|syslog|unix_dgram|unix_stream
      filename: eve.json
      types:
        - alert
        - http:
            extended: yes
```

V případě že používáme Barnyard v kombinací se Snorby, nebo jiným monitorovacím rozhraním, je třeba nastavit třídu **unified2-alert**. Vhodné je nastavit i zobrazování HTTP logů pro zobrazování relací na webové úrovni. Povolením **pcap-log** logování, umožníme zaznamenávat výstupy ve formátu vhodném pro analýzu přes Wireshark. [8]

```
  - unified2-alert:
      enabled: yes
      filename: unified2.alert
  - http-log:
      enabled: yes
      filename: http.log
  - pcap-info:
      enabled: yes
  - pcap-log:
      enabled:  yes
      filename: log.pcap
      limit: 1000mb // velikost logu
      max-files: 2000// definováním aktivujeme režim ring buffer
  - drop:
      enabled: yes
      filename: drop.log
```

```
      append: yes
```

Nastavení pro režim IPS, v kterém bude protékat síťový provoz z jednoho rozhraní na druhé. Parametr **CLUSTER-TYPE** nastavuje load balancing mezi rozhraními podle toků nebo hashů. V našem případě máme nastaveno podle toků. Použití pro kernel vyšší než 3. 1. **COPY-MODE: IPS** nám zajistí kopírování provozu z jednoho rozhraní na druhé při respektování pravidla DROP, kdežto **TAP** přepošle pakety označené DROP nezahazuje, ale jen přeposílá z jednoho rozhraní na druhé. [8]

```
af-packet:
  - interface: eth0
    threads: 1
cluster-id: 99
    cluster-type: cluster_flow
    defrag: yes
    use-mmap: yes
    copy-mode: ips // IPS nebo TAP
    copy-iface: eth1 // rozhraní kde chceme data kopírovat
    buffer-size: 64535
```

Parametry pro druhé rozhraní jsou totožné, nutné je zachovat stejnou velikost bufferu.

```
  - interface: eth1
    threads: 1
    cluster-id: 98
    cluster-type: cluster_flow
    defrag: yes
    buffer-size: 64535
    use-mmap: yes
    copy-mode: ips
    copy-iface: eth0
```

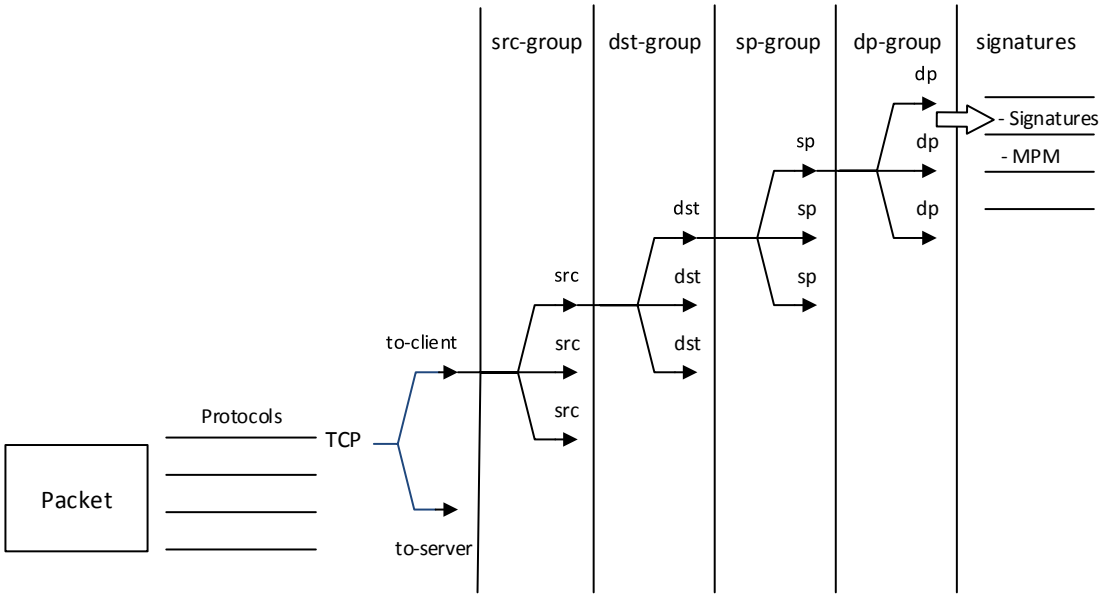
Další třídou pro nastavení je třída **DETECT-ENGINE**. Suricata pro analýzu používá signatury. V případě, kdy máme do suricaty importováno mnoho pravidel, bylo by neefektivní, kdyby se pakety porovnávaly se všemi pravidly. [8]

Proto existuje mechanizmus, který efektivně analyzuje pakety rozdělování do skupin a procházení mezi těmito skupinami funguje na principu stromovité struktury. Avšak ne každá signatura má svou skupinu a značné množství signatur takového typu může vést k vyšším nárokům na paměť stroje. [8]

Toto můžeme prostřednictvím parametru profile ovlivnit. Můžeme říct Suricatě, nechť třídí poctivě podle různých kritérií, nebo ne. Mimo to, můžeme ještě specifikovat tuto konfiguraci vlastním nastavením.

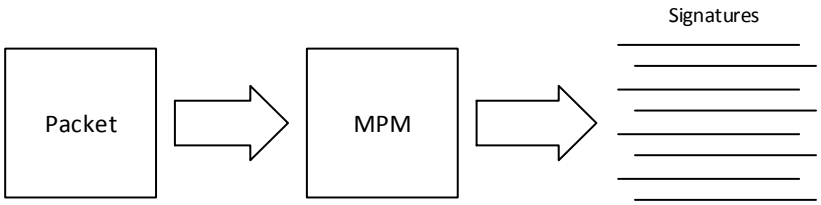
```
detect-engine:
  - profile: medium
  - custom-values:
      toclient-src-groups: 2
      toclient-dst-groups: 2
      toclient-sp-groups: 2
      toclient-dp-groups: 3
      toserver-src-groups: 2
      toserver-dst-groups: 4
      toserver-sp-groups: 2
      toserver-dp-groups: 25
  - sgh-mpm-context: auto
  - inspection-recursion-limit: 3000
```

Parametrem **SGM-MPM-CONTEXT** nastavíme, zda bude mít každá skupina vlastní více vzorkovou identifikaci MPM (single) nebo bude sdílená všemi skupinami (full). Hodnota **auto** vybírá podle definovaného algoritmu, Obrázek 3.3. [8]



Obrázek 3.3: Mechanizmus jádra rozdělování do skupin

Suricata ke své detekci používá speciální část nazývanou **MULTIPATERN MATCHER**. Tato funkce umožnuje detekovat signatury podle různých vzorků. [8] Při shodě různých vzorků může Suricata identifikovat přesně daný typ signatury, Obrázek 3.4. MPM používá pro svou činnost různé algoritmy.



Obrázek 3.4: Využití MPM při detekci

```
pattern-matcher:
  - b2gc:
      search-algo: B2gSearchBNDMq
      hash-size: low
      bf-size: medium
  - b2gm:
      search-algo: B2gSearchBNDMq
      hash-size: low
      bf-size: medium
  - b2g:
      search-algo: B2gSearchBNDMq
      hash-size: low
      bf-size: medium
  - b3g:
      search-algo: B3gSearchBNDMq
      hash-size: low
      bf-size: medium
  - wumanber:
      hash-size: low
      bf-size: medium
```

MEMCAP definuje prostor pro alokaci paměti v jádře. **HASH-SIZE** definuje velikost hash používaného k identifikaci jednotlivého toku v jádře. Pokud počet jednotlivých toků v pamětí převýší tuto mez, **EMERGENCY-RECOVERY** pozdrží vytváření nových toků. Tuto

činnost provádí v intervalech definovaných v sekundách. **PREALLOC** umožnuje zvýšení efektivity předběžným načtením určitého množství toků. [8] V další části můžeme definovat různé datové toky.

```
flow:
  memcap: 64mb
  hash-size: 65536
  prealloc: 10000
  emergency-recovery: 30
flow-timeouts:
  default:
    new: 30
    established: 300
    closed: 0
    emergency-new: 10
    emergency-established: 100
    emergency-closed: 0
  tcp:
    new: 60
    established: 3600
    closed: 120
    emergency-new: 10
    emergency-established: 300
    emergency-closed: 20
  udp:
    new: 30
    established: 300
    emergency-new: 10
    emergency-established: 100
  icmp:
    new: 30
    established: 300
    emergency-new: 10
```

```
    emergency-established: 100
```

Nastavení detekčního jádra. Zde probíhá proces zpracovávání TCP toku. Parametrem **CHECKSUM-VALIDATION** nastavíme citlivější analýzu pro pakety s korektním CRC záznamem. V případě povolení této volby budou pakety s chybnou CRC logovány, v opačném případě budou zahozeny. Zamítnutí této volby můžeme zvýšit riziko špatné detekce potenciálních útoků. Řešení spočívá v úpravě parametrů síťové karty např. pomocí nástroje ethtool. **INLINE** je parametr pro aktivaci inline v režimu IPS. [8]

Třída **REASEMBLY** definuje vlastnosti pro skládání toků. Nastavení parametru **DEPTH** definujeme, po jakých částech se bude skládat jednotlivý tok. Parametry **TOSERVER-CHUNK-SIZE** a **TOCLIENT-CHUNK-SIZE** definují minimální velikost pro inspekci raw streamu. [8]

```
stream:
  memcap: 32mb
  checksum-validation: yes
  inline: yes
  reassembly:
    memcap: 128mb
    depth: 1mb
    toserver-chunk-size: 2560
    toclient-chunk-size: 2560
    randomize-chunk-size: yes
```

Třída pro nadefinování výchozí cesty pro logování. V našem případě jsme využívali hlavně záznamy typu fast a http log.

```
outputs:
  - console:
      enabled: yes
  - file:
      enabled: no
      filename: /var/log/suricata.log
```

V další fázi popíšeme konfiguraci alternativního paketového analyzátoru PF-Ring. Povoluje segmentaci paketů do jednotlivých vláken, nastavíme parametrem **THREADS**. Další výhodou je analýza paketů na nižší úrovní než u ostatních paketových analyzátorů, které využívají formátu pcap. **CLUSTER-ID** rozděluje toky do stejně velkých front. Vlákna, které se podílí na procesu, musí mít stejné **CLUSTER-ID**. V další části nastavíme výchozí adresář pro naše pravidla a názvy jednotlivých pravidel.

```
pfring:
  - interface: eth1
    threads: 1
    cluster-id: 99
default-rule-path: /etc/suricata/rules
rule-files:
 - botcc.rules
 - ciarmy.rules
     …
```

V souboru **CLASSIFICATION** se nachází definice struktury jednotlivých pravidel. V souboru reference nalezneme odkazy na weby, které souvisí s problematikou IDS/IPS. V další části konfigurujeme adresové rozsahy sítě. **ACTION-ORDER** definuje jednotlivé akce pro manipulaci s pakety. Velikostí parametru **ASN1-MAX-FRAMES** nastavujeme maximální počet rámců pro inspekci těchto datových struktur. [8]

```
classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
address-groups:
    HOME_NET: "[1.0.0.0/24]"
    EXTERNAL_NET: any
    HTTP_SERVERS: "$HOME_NET"
     …
action-order:
  - pass
  - drop
  - reject
  - alert
asn1-max-frames: 256
```

Aktivací parametru **RULES-FAST-PATERN** nastavíme potlačení pro více vzorovou identifikaci. **RULES** ve stavu „*yes*“ nastaví výpis hlášení pro každé pravidlo. V další částí nastavujeme knihovnu *libhtp*, která zodpovídá za zpracování http relací. Často se stává, že jsou tyto zprávy objemné a tak je důležité nastavit limity, protože mají vliv na výkonnost IDS. Nastavení parametrů **REQUEST A RESPONSE BODY LIMIT** na hodnotu 0 Suricata zpracuje celou zprávu. [8]

```
engine-analysis:
  rules-fast-pattern: yes
  rules: yes
libhtp:
         default-config:
           personality: IDS
           request-body-limit: 3072
           response-body-limit: 3072
           request-body-minimal-inspect-size: 32kb
           request-body-inspect-window: 4kb
           response-body-minimal-inspect-size: 32kb
           response-body-inspect-window: 4kb
```

Nastavení profilování pro každý paket nebo pravidlo. Každý n-tý paket je profilován. Při detekci je nutné si uvědomit, že každé pravidlo je různě složité a časté profilování pro složitá pravidla může vézt k snížení výkonu Suricaty. Proto je ve výchozím stavu profilování vypnuto a pro použití je nutné Suricatu zkompilovat s tímto parametrem **--enable-profiling**. Na konci každé relace Suricata vypíše statistiky. Podle parametru **SORT** definujeme, jak se bude výstupní záznam třídit. Parametr **LIMIT** definuje množství zobrazovaných signatur.

```
  rules:
    enabled: yes
    filename: rule_perf.log
    append: yes
    sort: avgticks
    limit: 100
```

3.3 Volitelná instalace monitorovací služby SNORBY

Jelikož nám Suricata zobrazuje výstupy do textových souborů, je možnost doinstalovat tuto webovou službu, který bude zobrazovat výstupy graficky.

```
# sudo apt-get install gcc g++ build-essential libssl-dev
libreadline6-dev zlib1g-dev linux-headers-generic libsqlite3-dev
libxslt-dev libxml2-dev imagemagick git-core libmysqlclient-dev
mysql-server libmagickwand-dev default-jre ruby1.9.3
```

Dále nainstalujeme Ruby, které je nutné pro běh Snorby.

```
# sudo gem install thor i18n bundler tzinfo builder memcache-
client rack rack-test erubis mail text-format rack-mount rails
sqlite3

# sudo git clone http://github.com/Snorby/snorby.git
/var/www/snorby

# cd /var/www/snorby

# sudo bundle update activesupport railties rails

# sudo gem install arel ezprint && sudo bundle install

# sudo bundle exec rake snorby:setup
```

Tímto jsme stáhli adresář Snorby s potřebnými soubory pro korektní běh do složky /var/www/snorby. V dalším kroku vytvoříme databázi pro Snorby a nastavíme konfigurační soubor pro databázi Snorby.

```
# mysql -u root -p

# create user 'snorbyuser'@'localhost' IDENTIFIED BY
'MojeHeslo';

# grant all privileges on snorby.* to 'snorbyuser'@'localhost'
with grant option;

# flush privileges;
```

Nyní editujeme konfigurační soubor databáze Snorby.

```
# cp /var/www/snorby/config/database.yml.example
/var/www/snorby/config/database.yml

# nano /var/www/snorby/config/database.yml

snorby: &snorby

  adapter: mysql

  username: root

  password: "MojeHeslo"

  host: localhost
```

V tomto kroku ověříme, zda MySQL naslouchá a okomentujeme v */etc/my.conf*

```
bind-address            = 127.0.0.1
```

3.3.1 Instalace a konfigurace webového serveru APACHE

V první fázi je třeba nainstalovat modul pro webový server, který dokáže spustit aplikace založené na Ruby.

```
# apt-get install apache2 apache2-prefork-dev libapr1-dev
libaprutil1-dev libopenssl-ruby libcurl4-openssl-dev
```

Editovat */etc/apache2/mods-available/passenger.load* (pokud neexistuje, vytvořit), pro nalezení tohoto souboru můžeme využít následující příkaz, který vám vypíše umístění souboru.

```
# find / -name "*mod_passenger*"
# nano /etc/apache2/mods-available/passenger.load
```

Do souboru *passenger.load* vložíme tyto řádky.

```
LoadModule passenger_module /var/lib/gems/1.9.1/gems/passenger-
4.0.41/buildout/apache2/mod_passenger.so

<IfModule mod_passenger.c>
PassengerRoot /var/lib/gems/1.9.1/gems/passenger-4.0.41
PassengerRuby /usr/bin/ruby
</IfModule>
```

Poté zaregistrujeme modul a spustíme Apache.

```
#a2enmod passenger
#a2enmod rewrite
#a2enmod ssl
#chmod 777 –R /var/www/snorby
```

Vytvoříme záznam „*snorby*“ v */etc/apache2/sites-available*

```
#nano /etc/apache2/sites-available
<VirtualHost *:80>
ServerAdmin webmaster@localhost
ServerName snorby
DocumentRoot /var/www/snorby/public

<Directory "/var/www/snorby/public">
AllowOverride all
Order deny,allow
Allow from all
Options -MultiViews
</Directory>

</VirtualHost>
```

Vytvoříme odkaz a restartujeme Apache.

```
#ln -s /etc/apache2/sites-available/snorby /etc/apache2/sites-enabled/snorby_config
#service apache2 restart
```

Spustíme webový prohlížeč a přihlásíme se podle těchto přihlašovacích údajů.

```
http://localhost
Login: snorby@snorby.org
Password: snorby
```

Pokud budeme chtít využít exportování záznamů ze Snorby do formátu pdf, bude třeba nainstalovat program Wkhtmltopdf. Program můžeme stáhnout z oficiálních stránek pro danou distribuci a nainstalovat, nebo pomocí aptitude. Je nutné však ověřit správné cesty Wkhmtltopdf v konfiguračním souboru */var/www/snorby/config/snorby_config.yml*, popřípadě editovat.

```
# wkhtmltopdf: /usr/bin/wkhtmltopdf
```

Na webové stránce v Administration je třeba zapnout worker, jinak se nebudou sbírat data. Pokud však z nějakého důvodu worker nepůjde zapnout, musíme tak učinit ručně z příkazové řádky.

```
# cd /var/www/snorby
# rails console
# Snorby::Worker.stop
# Snorby::Jobs.clear_cache
# Snorby::Worker.start
# exit
```

3.3.2 Instalace Barnyard2

```
# apt-get install mysql-client libmysqlclient-dev libprelude-dev
# git clone https://github.com/firnsy/barnyard2.git
# cd barnyard2 && ./autogen.sh
# ./configure –-enable-debug --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu/
# make && make install
```

Pokud budete mít problém s kompilací Barnyardu2 jako v mém případě, kdy mi chyběla knihovna *dnet.h*, je ji třeba nalinkovat z jiného adresáře a poté kompilace proběhne.

```
# ln -s /usr/include/dumbnet.h dnet.h
```

Dále zkopírujeme soubor barnyard2.conf do /etc/suricata

```
# cd etc && cp barnyard2.conf /etc/suricata
```

Editujeme */etc/suricata/barnyard2.conf* na

```
config reference_file: /etc/suricata/reference.config
config classification_file: /etc/suricata/classification.config
config gen_file: /etc/suricata/rules/gen-msg.map
config sid_file: /etc/suricata/rules/sid-msg.map
```

Nastavíme výstup do databáze Snorby a další nezbytné parametry:

```
output database: log, mysql, user=root password=MojeHeslo
dbname=snorby host=localhost
output alert_fast
config logdir: /var/log/suricata
config waldo_file: /var/log/suricata/barnyard2.waldo
config hostname: suricata-ids
config interface: eth0
config daemon
```

Spustíme barnyard2. Je vhodné nespouštět prvně v režimu démona, v případě nějakých chyb (bez parametru –D).

```
#barnyard2 -c /etc/suricata/barnyard2.conf -d /var/log/suricata
-f unified2.alert –D
```

3.4 Modifikace parametrů virtuálních strojů

3.4.1 Úprava síťových karet na IDS / IPS

Tato modifikace byla nezbytná pro korektní detekci příchozí komunikace. Síťové rozhraní, přes které procházela komunikace, bylo nutné upravit. K tomuto účelu nám posloužil linuxový níže definovaný příkaz.

```
# for i in rx tx sg tso ufo gso gro lro; do ethtool -K eth0 $i
off; done
```

Příkaz je nutné aplikovat na všechna rozhraní účastnící se analýzy, především při testování v režimu IPS. Následujícím příkazem si ověříme provedené změny. Výpis níže zobrazuje korektně nastavené parametry síťového rozhraní.

```
# sudo ethtool -k eth0
Offload parameters for eth0:
rx-checksumming: off
```

```
tx-checksumming: off
scatter-gather: off
tcp-segmentation-offload: off
udp-fragmentation-offload: off
generic-segmentation-offload: off
generic-receive-offload: on
large-receive-offload: off
rx-vlan-offload: on
tx-vlan-offload: on
ntuple-filters: off
receive-hashing: off
```

Nepoužitím tohoto příkazu může vést k nekorektní detekci. Suricata může pak většinu síťového provozu detekovat takto:

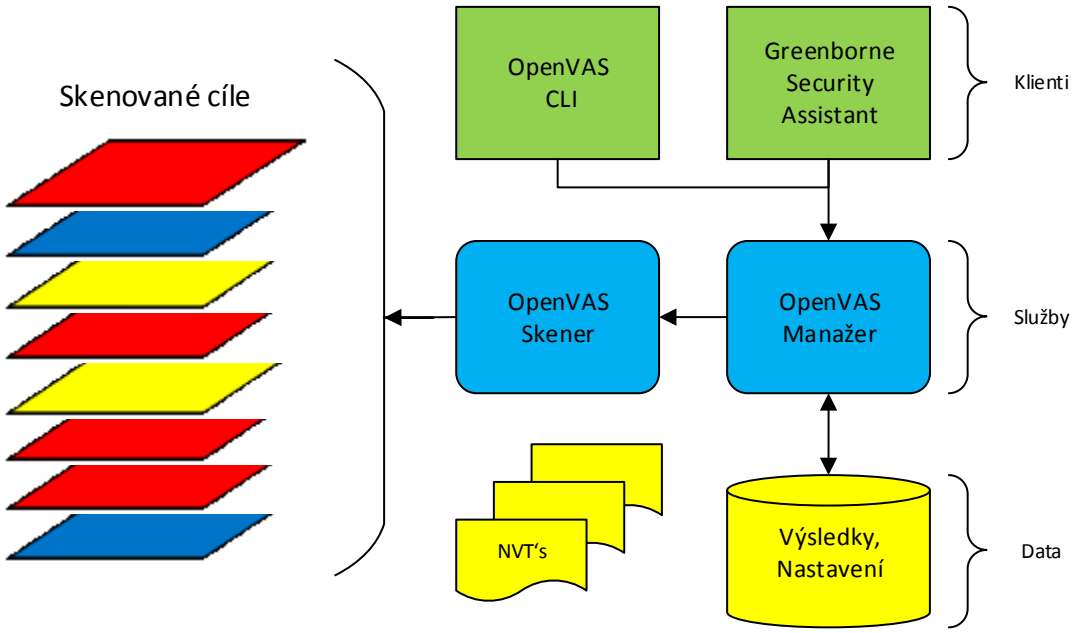
```
SURICATA TCPv4 invalid checksum [**] [Classification: (null)]
[Priority: 3]
```

4 Funkční analýza a výkonnostní testování IDS Suricata při nasazení v testovací topologii.

Pro testovací účely jsem využil penetračního frameworku OpenVAS (Open Vulnerability Assessment System). Tento framework zahrnuje v sobě komplexní testy, které administrátor spouští přes webové rozhraní. Penetrační testy jsou aktualizovány průběžně a v současné době existuje zhruba 35000 NVT testů pro OpenVAS. Projekt financují organizace, jejich zájmem je zlepšovat bezpečnost v oblasti počítačových sítí a systémů.

4.1 Architektura OpenVAS

Z pohledu architektury, která běží pod SSL knihovnou je za jádro OpenVAS považován blok skener. Jeho úkolem je spouštět penetrační testy, které bývají denně aktualizovány prostřednictvím OpenVAS NVT nebo komerční služby. [9]



Obrázek 4.1: Architektura frameworku OpenVAS

Úkolem Manažera je centrální správa, která sjednocuje testy různých typů pod jednu testovací základnu. Manažer řídí skener pomocí protokolu OTP a dále nabízí protokol založený na XML, nazývající se OMP. Všechno řízení je implementováno v manažeru a tak se nabízí možnost implementace různých klientů, které se budou chovat konzistentně například s ohledem na filtrování nebo třízení výsledků. Dále manažer řídí SQL databázi kde je uložena konfigurace a výsledky všech testů. Jedním z úkolů manažera je také řízení uživatelského prostředí, přístupy do něj a práva. [9]

Existují různí OMP klienti. Webová služba Greenbone Security Assisstant je podpůrnou službou nabízející uživatelské prostředí pro testování. Tato služba používá pro zobrazení OMP do HTML transformaci XSL. [9]

OpenVAS CLI nabízí vytvoření spouštěcích procesů k ovládání OpenVAS manažera. Součástí je i balík pro NAGIOS. Pomocí protokolu OTP můžeme řídit spouštění testů. [9]

4.2 Instalace OpenVAS v OS Kali

V operačním systému Kali OS je OpenVAS předinstalovaný, je pouze nutné zapnout dané služby. Pomocí následujících příkazů spustíme nezbytné služby a spustíme webovou službu GSA. V devíti krocích ověří následující příkaz instalaci OpenVAS na stroji.

```
# ./openvas-check-setup
```

V případě, že tento skript nahlásí že OpenVAS není korektně nainstalován, spustíme další příkaz. Tento skript nám spustí potřebné služby a nainstaluje na lokální stroj OpenVAS včetně webového rozhraní GSA. Při této instalaci nastavíme rovněž heslo k webové administraci pro uživatele admin. Po úspěšné instalaci následujícího skriptu přejdeme webovým prohlížečem na http://localhost:9392.

```
# ./openvas-setup
```

Pokud budeme chtít zaktualizovat databázi NVT, využijeme příkazu:

```
# ./openvas-feed-update
```

4.3 Testování IDS pomocí penetračního nástroje OpenVAS

V testovacím prostředí OpenVAS jsem vytvořil vlastní testovací scénář. Tento scénář se skládá z 25 testů různých kategorií. Testy byly vybrány svobodně se snahou vybrat typického zástupce z dané kategorie, Obrázek 4.2.

Family	NVTs selected	Trend	Actions
Brute force attacks	1 of 8		
CISCO	1 of 20		
Databases	1 of 136		
Default Accounts	1 of 77		
Denial of Service	3 of 921		
Gain a shell remotely	1 of 92		
Malware	3 of 42		
Product detection	1 of 443		
SMTP problems	2 of 48		
SNMP	1 of 6		
Web Servers	2 of 240		
Web application abuses	5 of 3226		
Windows	1 of 141		
Windows : Microsoft Bulletins	2 of 766		
Total: 14	25 of 6166 in selected families of 38064 in total		

Obrázek 4.2: Jednotlivé testovací kategorie v prostředí OpenVAS

Tabulka 4.1 zobrazuje výčet všech jednotlivých testů, které byly použity při testování. Testy byly vybírány s ohledem na aktuální situaci v oblasti bezpečnosti počítačových sítí. [10, 11] Příkladem můžou být útoky typu DDoS, které jsou na denním pořádku. Cílem hackerských útoků bývají i špatně zabezpečené databáze, weby a různé další služby. Testovány byly i produkty firmy Microsoft, jako je například zranitelnost prohlížeče Internet Explorer, .NET nebo ASP.NET aplikací. Tyto testy byly použity v režimu IDS a IPS. V další kapitole budou znázorněny grafické výstupy z monitorovacího prostředí Snorby.

Family	Type of attack
Brute force attacks	ncrack: SSH
CISCO	Cisco IOS XR Software Fragmented Packets Processing Denial of Service Vulnerability
Databases	MySQL Authentication Bypass
Default Accounts	MySQL weak password
Denial of Service	Google Chrome Denial of Service Vulnerability - April 13 (Windows)
	Mozilla Firefox Denial Of Service Vulnerability - Sep09 (Win)
	ping of death
Gain a shell remotely	HTTP Cookie overflow
Malware	64-bit Debian Linux Rootkit with nginx Doing iFrame Injection
	Bugbear.B web backdoor
	Trojan horses
Product detection	MySQL/MariaDB Detection
SMTP problems	Generic SMTP overflows
	Sendmail remote header buffer overflow
SNMP	An SNMP Agent is running
Web Servers	Apache 2.0.39 Win32 directory traversal
	Microsoft IIS ASP Stack Based Buffer Overflow Vulnerability
Web application abuses	/cgi-bin directory browsable ?
	Joomla! JooProperty Component SQL Injection and Cross Site Scripting Vulnerabilities
	WordPress Pretty Link Lite Plugin SQL Injection And XSS Vulnerabilities
	XWiki Enterprise Unspecified SQL Injection and XSS Vulnerabilities
	phpMyAdmin Unspecified SQL Injection and Cross Site Scripting Vulnerabilities
Windows	Microsoft .NET 'ASP.NET' Cross-Site Scripting vulnerability
Windows : Microsoft Bulletins	Cumulative Security Update for Internet Explorer (972260)
	Microsoft Windows Active Directory Denial of Service Vulnerability (973309)

Tabulka 4.1: Jednotlivé testy použité v testování

4.4 Testování IDS pomocí penetračního nástroje Nessus

Další nástroj umožňující penetrační testování je Nessus. Nástroj podobně jak OpenVAS umožnuje testovat zranitelnost systémů. Mnoho penetračních testů obsahující Nessus je totožných jako v OpenVAS, avšak ne všechny. Testovací scénář byl vybrán tak, aby se nejvíce podobal tomu pro předchozímu pro OpenVAS. Nakonec se podařilo sestavit totožný scénář až na jeden test pro XWIKI, který byl nahrazen MediaWIKI.

4.5 Využití pravidel VRT SNORT

Součástí práce bylo ověřit, zda se dá využít VRT pravidel používané konkurenčním IDS systémem SNORT. Dokumentace uvádí, že je možné tyto pravidla aplikovat i pro Suricatu. Odkaz na pravidla jsem tedy vložil do konfiguračního souboru programu OINKMASTER, určeného pro automatické stahování pravidel ET, ET PRO a VRT pro Suricatu. Program však během instalace pravidel zahlásil chybové hlášky tohoto typu:

```
# Checking flowbits dependencies... problems found:
```

Program instalační akci dokončil a nakopíroval do cílové cesty pravidla, avšak je nenalinkoval do konfiguračního souboru Suricaty. Ručně jsem pravidla nalinkoval, ale SURICATA při zapnutí hlásila chyby při parsování signatur, viz níže.

```
18/4/2015 -- 23:38:03 - <Info> - 119 rule files processed. 4360
rules successfully loaded, 9727 rules failed

18/4/2015 -- 23:38:03 - <Info> - 4360 signatures processed. 0
are IP-only rules, 2169 are inspecting packet payload, 2368
inspect application layer, 0 are decoder event only
```

SURICATA byla spuštěna se 4360 pravidly, 9827 pravidel však načíst nemohla. Můžeme pozorovat, že kompatibilita se SNORT VRT pravidly není až tak vyladěná. Takto aktivována SURICATA s VRT pravidly se podrobila stejnému testu jako v předchozí kapitole, avšak při následném zkoumání logů bylo zjištěno, že žádné hrozby nedetekovala a absolutně našemu testu nevyhověla.

5 Grafické zpracování výsledků testování.

V této kapitole se podíváme na grafické výsledky testování pomocí frameworku OpenVAS, zejména na reakce IDS Suricata. Pro detekci anomálií a hrozeb využívala Suricata IDS pravidla společnosti Emerging Threats.

5.1 Testované služby

Tabulka 5.1 uvádí seznam použitých útoků a jejich detekce Suricatou. Jelikož útoky vycházely z testovacího frameworku OpenVAS, většina z nich byla takto detekována. Sloupec vlevo označuje typ útoků, prostřední sloupec říká, zda byl daný útok detekován Suricatou a třetí sloupec vpravo oznamuje, zda byl při daném typu útoku zaznamenán provoz na http úrovni. Jak můžeme vidět z tabulky, SURICATA při detekci obstála na výbornou, neboť detekovala každý test určitým typem signatury.

Type of attack	Detection by Suricata fast.log	Deep Analyse http.log
ncrack: SSH	YES	NO
Cisco IOS XR Software Fragmented Packets Processing Denial of Service Vulnerability	YES	YES
MySQL Authentication Bypass	YES	NO
MySQL weak password	YES	NO
Google Chrome Denial of Service Vulnerability - April 13 (Windows)	YES	NO
Mozilla Firefox Denial Of Service Vulnerability - Sep09 (Win)	YES	NO
ping of death	YES	NO
HTTP Cookie overflow	YES	YES
64-bit Debian Linux Rootkit with nginx Doing iFrame Injection	YES	NO
Bugbear.B web backdoor	YES	YES
Trojan horses	YES	NO
MySQL/MariaDB Detection	YES	
Generic SMTP overflows	YES	NO
Sendmail remote header buffer overflow	YES	NO
An SNMP Agent is running	YES	NO
Apache 2.0.39 Win32 directory traversal	YES	YES
Microsoft IIS ASP Stack Based Buffer Overflow Vulnerability	YES	YES
/cgi-bin directory browsable ?	YES	YES
Joomla! JooProperty Component SQL Injection and Cross Site Scripting Vulnerabilities	YES	YES

WordPress Pretty Link Lite Plugin SQL Injection And XSS Vulnerabilities	YES	YES
XWiki Enterprise Unspecified SQL Injection and XSS Vulnerabilities	YES	YES
phpMyAdmin Unspecified SQL Injection and Cross Site Scripting Vulnerabilities	YES	YES
Microsoft .NET 'ASP.NET' Cross-Site Scripting vulnerability	YES	YES
Cumulative Security Update for Internet Explorer (972260)	YES	NO
Microsoft Windows Active Directory Denial of Service Vulnerability (973309)	YES	NO

Tabulka 5.1: Tabulka všech typů útoků použitých během testování

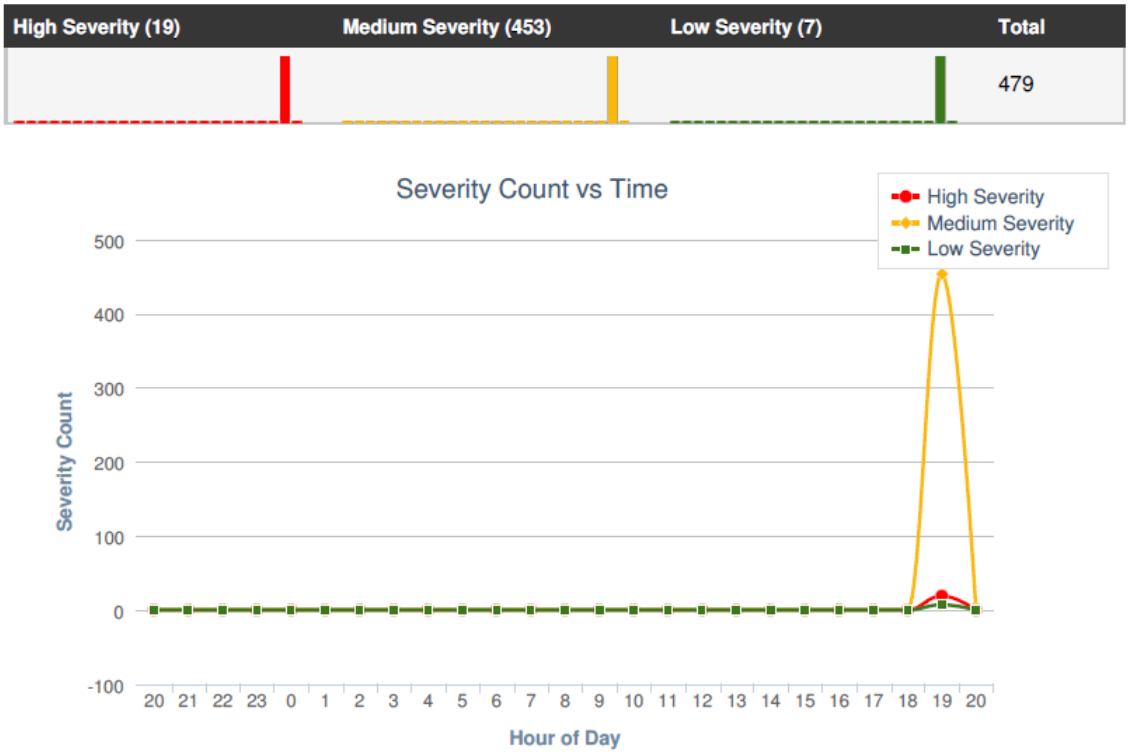
5.2 Grafické výstupy ze SNORBY

Snorby nabízí exportování událostí do formátu pdf. Obrázek 5.1 níže znázorňuje část výstupu obsaženého v pdf. Většinu detekováních útoku Suricata identifikovala jako OpenVAS SCAN, což je nejčetnější signatura, jak vidíme na obrázku. Můžeme si všimnout, že je signatur méně než počet testů, jelikož mnoho útoků shlukuje pod jednu signaturu. Pravidla pro Suricatu byl pouze staženy a naimportovány. V žádném případě nebyli nijak modifikovány.

Signature Name	Percentage	Event Count
ET SCAN OpenVAS User-Agent Inbound	75.99%	364
GPL WEB_SERVER .htaccess access	8.77%	42
GPL SNMP private access udp	2.51%	12
ET SNMP Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden ...	2.51%	12
GPL SNMP public access udp	2.3%	11
GPL SMTP vrfy root	1.88%	9
GPL SMTP expn root	1.67%	8
ET SNMP Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden ...	1.25%	6
GPL NETBIOS SMB-DS Session Setup NTMLSSP unicode asn1 overflow...	0.63%	3
GPL WEB_SERVER 403 Forbidden	0.63%	3
ET POLICY Suspicious inbound to mySQL port 3306	0.42%	2
SURICATA ICMPv4 unknown code	0.42%	2
SURICATA UDPv4 invalid checksum	0.42%	2
GPL SMTP RCPT TO overflow	0.21%	1
ET POLICY Suspicious inbound to MSSQL port 1433	0.21%	1
ET SCAN Potential SSH Scan	0.21%	1

Obrázek 5.1: Seznam odhalených signatur v testu

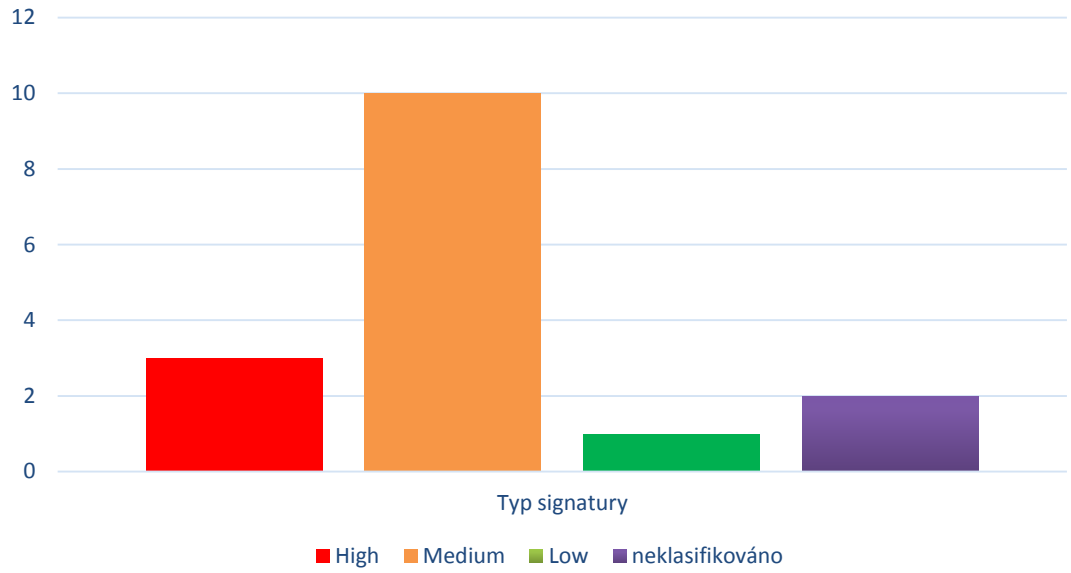
Na dalším obrázku 5.3 můžeme vidět reakci Suricaty po prvním penetračním testu. Ze 479 detekovaných událostí je 19 vysoce kritických, 453 středně kritických a 7 nízko kritických. Snorby jednotlivé závažnosti reprezentuje barvami. Čím vyšší je závažnost útoků, tím je větší šance na získání kontroly nad systémem. Je to určitý způsob klasifikace nebezpečnosti útoků. Snorby bohužel nenabízí zobrazování grafů s měřítkem menším než je posledních 24 hodin po hodinových intervalech.



Obrázek 5.2: Počet detekovaných signatur podle závažnosti

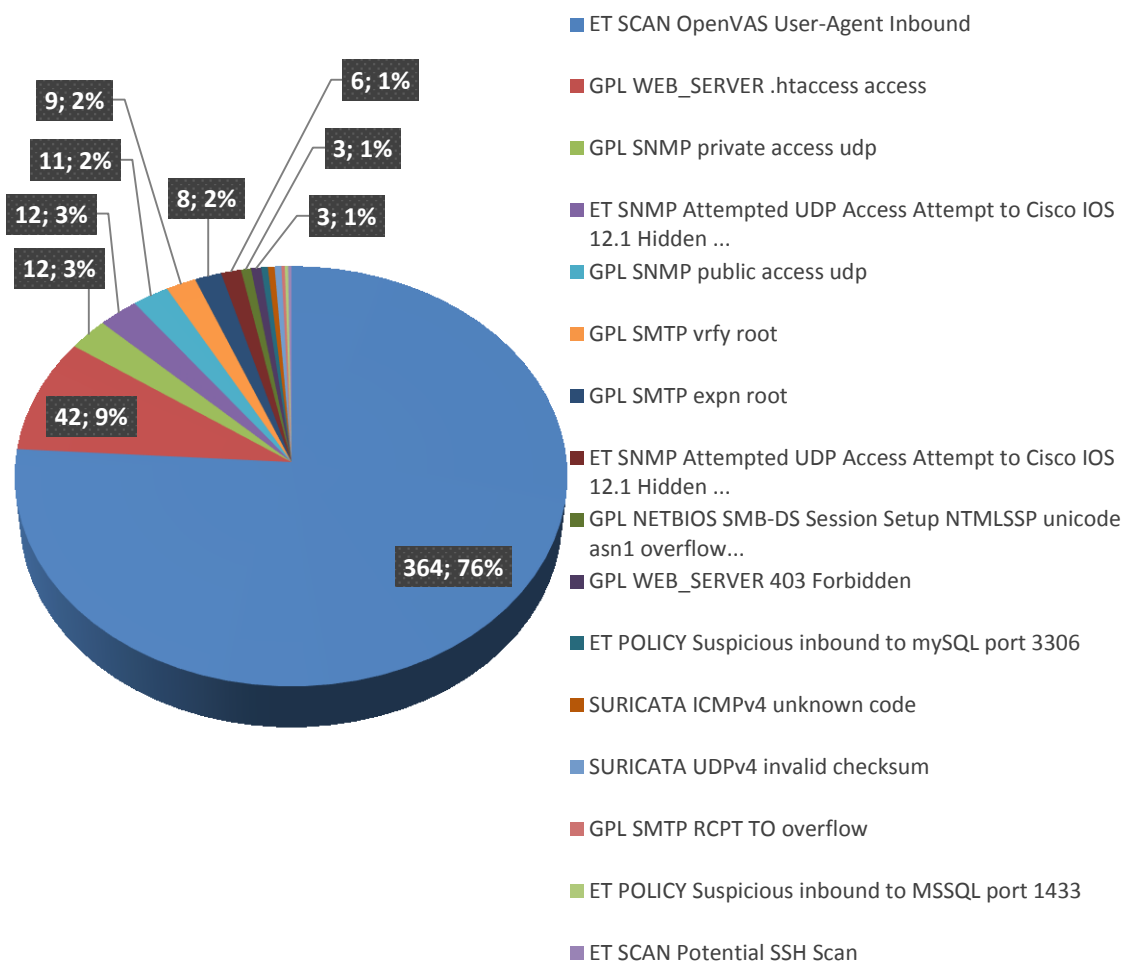
5.3 Grafická analýza útoků - OpenVAS

Předchozí graf ukazuje celkový počet signatur zahrnující i ty, které se v testování opakují. Následující Graf 5.1 zobrazuje počet unikátních signatur detekovaných při testování rozdělených podle závažnosti. Unikátních signatur bylo celkem 16, z toho 3 vysoce závažné, 10 středně závažných, 1 málo závažná. Dvě signatury byly neklasifikované. To jsou především ty, u kterých není přesně dán charakter a cílový dopad na systém. Klasifikace signatur je nastavena podle toho jak jsou útoky nebezpečné a co můžou způsobit. V testu byla jako vysoce závažná například signatura útoku na CISCO směrovače, která měla za cíl získat administrátorský přístup ke směrovači. Jako další typický kandidát pro středně závažnou signaturu, bylo například skenování portů. S nejnižší závažnosti byla označena signatura, jejíž cílem bylo skenování služeb Microsoft.



Graf 5.1: Počet unikátních signatur rozdělených podle závažnosti

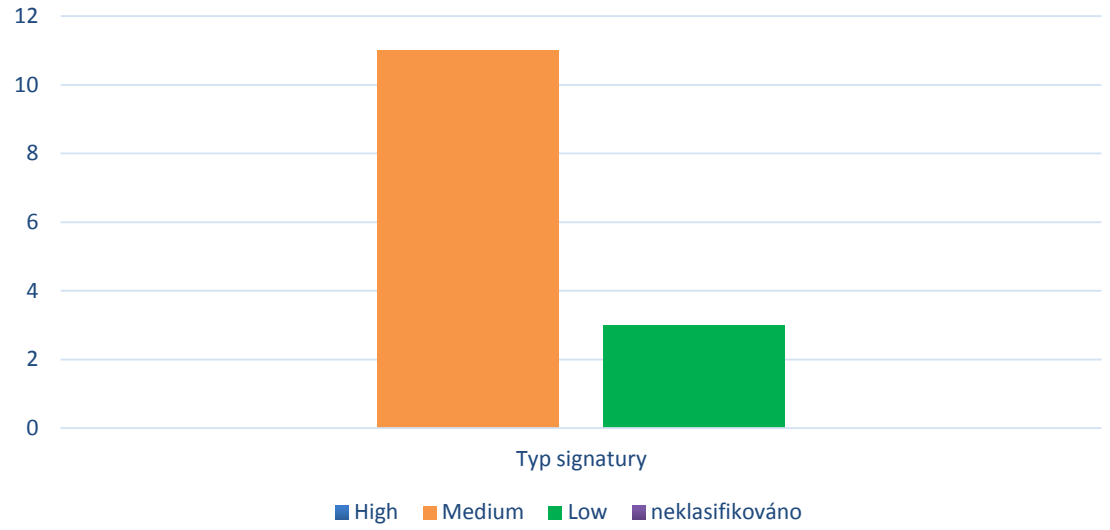
Graf 5.2 zobrazuje množství signatur nalezených při testování v IDS režimu. Jak je vidět z grafu, 76 % všech útoků detekovalo jako OpenVAS agenta. Každý test pro OpenVAS je napsaný různě, ale většina z nich má testuje cíl podobným způsobem. Pro analýzu dané služby používá například SNMP protokol, kterým si ověří verzi, a poté testuje. Někdy však testuje přímo.



Graf 5.2: Procento detekovaných signatur v IDS režimu pro OpenVAS

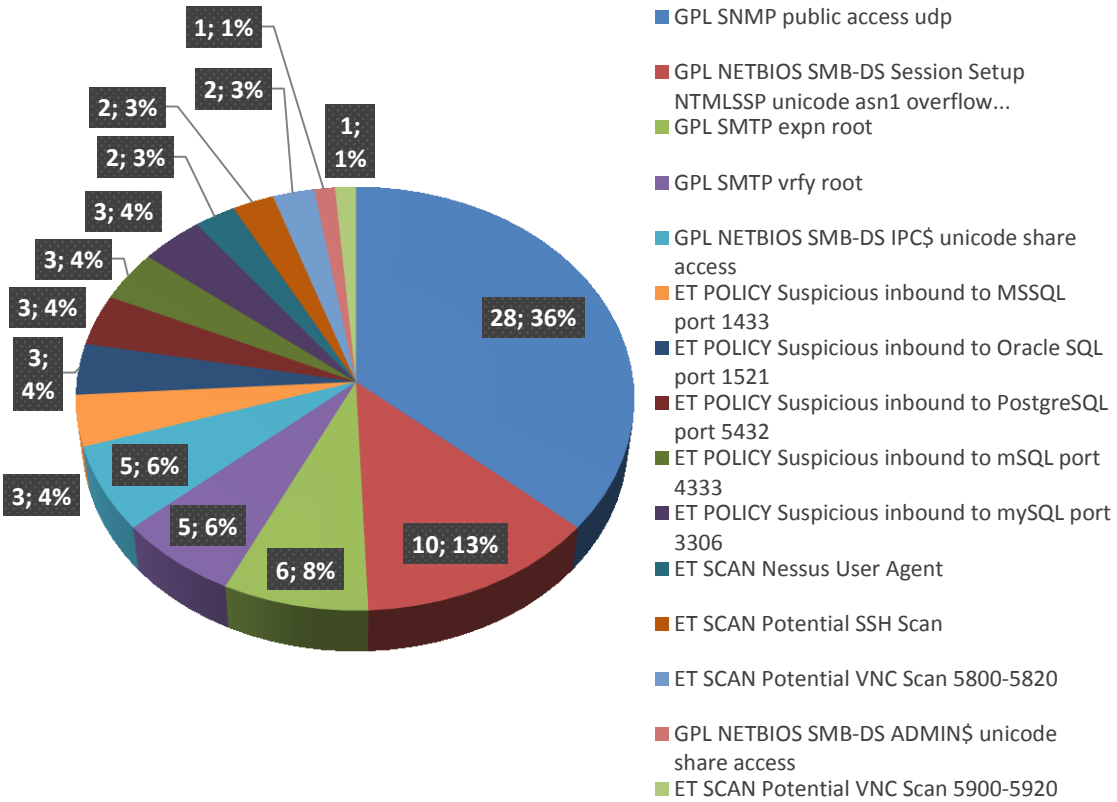
5.4 Grafická analýza útoků – Nessus

Při testování pomocí penetračního frameworku Nessus byly detekovány signatury pouze druhého a třetího stupně závažnosti. Chyběla zde signatura na zranitelnost CISCO směrovačů, nicméně jako OpenVAS i Nessus používají podobný mechanizmus inicializace testování, a tak byl při testování pouze této signatury, zaznamenám podezřelý síťový provoz a Nessus jej klasifikoval jako signaturu 2. stupně, Graf 5.3.



Graf 5.3: Počet unikátních signatur rozdělených podle závažnosti

Z grafu 5.4 vidíme, jak značně ubylo signatur typu User Agent oproti testování pomocí OpenVAS. Suricata zde detekuje Nessus agenta pouze v 36% detekce, kdežto při testování pomocí OpenVAS to bylo 76%. I tak zde přibyly další signatury související s úvodní fázi, které používá při svém testování Nessus.



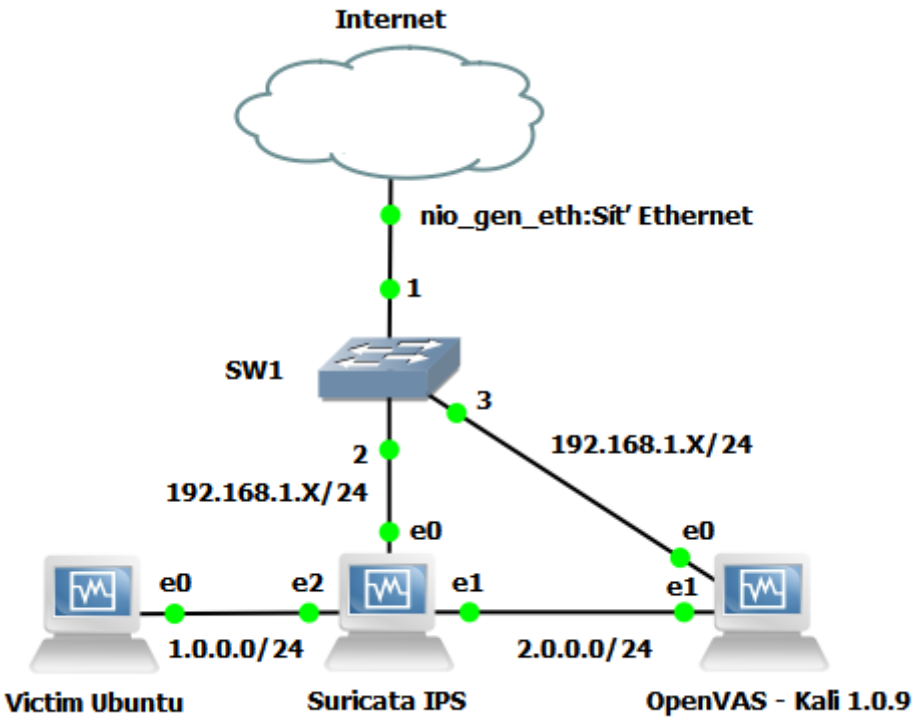
Graf 5.4: Procento detekovaných signatur v IDS režimu pro Nessus

6 Na základě získaných výsledků - návrh a praktická implementace skriptů pro dynamické nasazení bezpečnostních pravidel v ohrožených uzlech sítě.

Na základě předchozího testovacího schématu vytvořeného díky penetračnímu frameworku OpenVAS, byla Suricata otestována také v režimu IPS. Sjednocené testování umožnilo důkladně rozlišit výsledky režimu IDS oproti IPS. V následující kapitole bude rozebrán postup testování analýza útoků včetně výsledků a výsledných skriptů pro aktivní prevenci útoku v síti.

6.1 Testovací topologie

Testovací topologie musela být upravena, neboť oproti režimu IDS, se v režimu IPS chová SURICATA jako brána. Ve virtualizovaném prostředí programu GNS3 jsem proto vytvořil vhodnou topologii, Obrázek 6.1.



Obrázek 6.1: Testovací Topologie pro IPS režim

Při podrobném nahlédnutí do topologie, si musíme uvědomit, že bylo nutné správně nastavit správné směrovací cesty do příslušných sítí. To jsme učinili pomocí příkazu route.

6.2 Nastavení Suricaty v režimu IPS

Suricata nabízí efektivnější řešení za pomocí knihovny libnet. Díky naimplementované třídě v této knihovně je možné si vytvořit v IPTABLES prostor, na který se bude provoz zasílat. Takto vytvořený prostor se nazývá NFQUEUE. Místo předchozí úpravy IPTABLES nám stačí použít následujícího příkazu.

```
# sudo iptables -I FORWARD -j NFQUEUE
```

Suricatu v režimu IPS následně spustíme s tímto parametrem

```
# suricata –c /etc/suricata/suricata.yaml –q 0
```

V momentě, kdy spustíme Suricatu v tomto režimu, se vytvoří propojení mezi síťovými rozhraními a datový provoz může procházet.

6.3 Testovaní v režimu IPS

Pro testování v režimu IPS bylo nutné provést určité změny. Všechna pravidla stažená z Emerging Threads jsou pouze pro detekci a tak je nutné je upravit. Je však velice pracné upravovat ručně každé pravidlo a přepisovat jeho výchozí akci z ALERT na DROP nebo REJECT. Z tohoto důvodu jsem vytvořil skript, který po instalaci suricaty připraví suricatu na IPS režim. Níže bude skript vypsán a vysvětlen.

Inicializace skriptu

```
#!/bin/bash
```

Zavedení výchozích cest pro adresáře

```
DIRECTORY="/etc/suricata/rulesIPS/"
FILE="/etc/suricata/suricataIPS.yaml"
OLDPATH="/etc/suricata/rules"
NEWPATH="/etc/suricata/rulesIPS"
```

Podmínka, v které se vytvoří složka rulesIPS v /etc/suricata/ pouze v případě když neexistuje.

```
if [ -d "$DIRECTORY" ]; then
  echo "Directory ""$DIRECTORY" "exists"
else
  echo "Creating folder..."
  mkdir "$DIRECTORY"
fi
```

Zkopírování všech pravidel ve výchozí složce *rules* do nově vytvořené složky *rulesIPS*. V této složce se přepíší všechny řetězce obsahující ALERT na DROP.

```
cp /etc/suricata/rules/*.* "$DIRECTORY"

sed -i 's/alert/drop/g' "$DIRECTORY"*.rules

sed -i 's/nodrop/noalert/g' "$DIRECTORY"*.rules
```

Další blok obsahuje logickou podmínku, která vytvoří v případě neexistence nový konfigurační soubor pro režim IPS, odvozený z výchozího konfiguračního souboru. Upraví v něm cestu k novým pravidlům.

```
if [ -e "$FILE" ]; then
  echo "File ""$FILE" "exists"
else
  echo "Creating configuration file for IPS mode -
suricataIPS.yaml..."
  cp /etc/suricata/suricata.yaml /etc/suricata/suricataIPS.yaml
  sed -i "s|$OLDPATH|$NEWPATH|g" "$FILE"
fi
```

Tento skript je vhodné spouštět pravidelných intervalech, ideálně hned po automatické aktualizaci standartních pravidel pro IDS režim. Skript je programován v bashi a je vhodné ho spouštět automaticky přes CRON.

6.4 Výsledky testování v režimu IPS

Oproti předchozímu testování v režimu IDS významně klesl počet událostí. Vysvětlením je to, že Suricata při nalezení signatury v počáteční fázi aktivně blokuje další pakety obsahující stejnou signaturu.

Další možností je, že jednotlivé testy v OpenVAS scénáři využívají SNMP protokol ke zjišťování verzí různých služeb na testovací straně. Jakmile podmínkám vyhovují, spustí kompletní test. Proto je události až o 352 méně než v režimu IDS. Obrázek 6.2 nám ukazuje počet a procento nalezených signatur v testu.

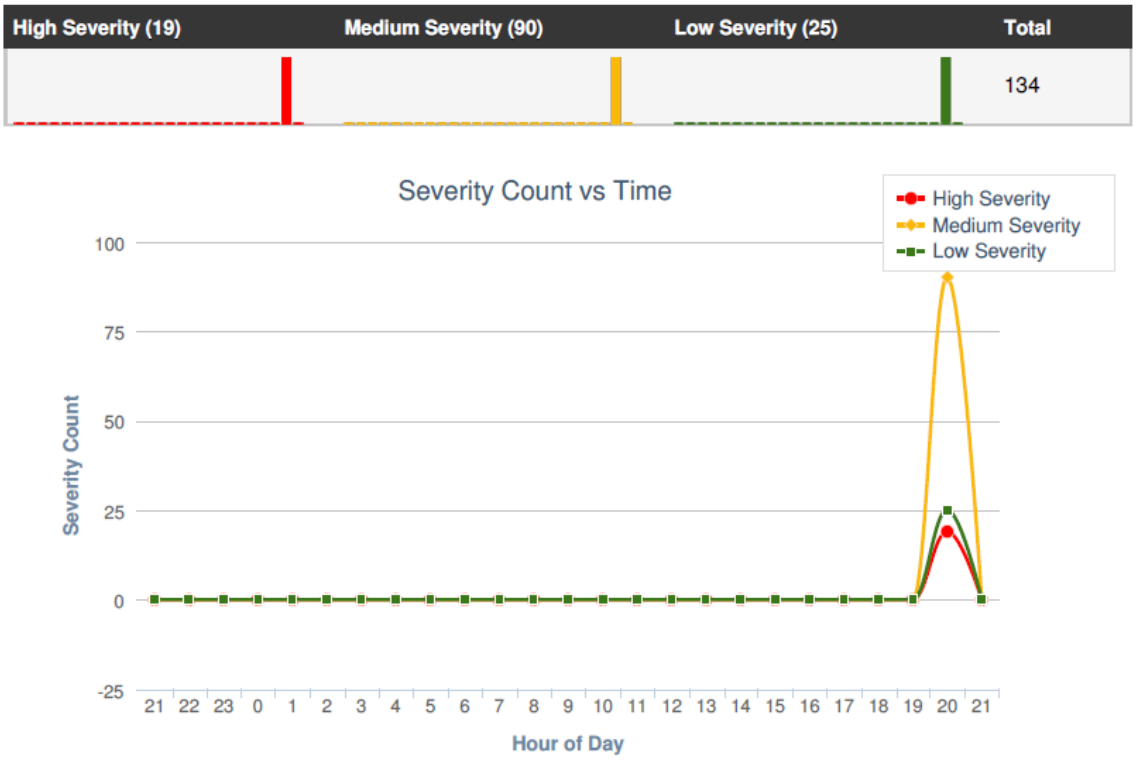
Signature Name	Percentage	Event Count
GPL SNMP public access udp	31.34%	42
ET SCAN OpenVAS User-Agent Inbound	18.66%	25
GPL NETBIOS SMB-DS Session Setup NTMLSSP unicode asn1 overflow...	15.67%	21
ET SNMP Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden ...	8.96%	12
GPL SNMP private access udp	8.96%	12
ET SNMP Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden ...	4.48%	6
ET POLICY Suspicious inbound to mySQL port 3306	3.73%	5
ET POLICY Suspicious inbound to MSSQL port 1433	2.99%	4
SURICATA UDPv4 invalid checksum	1.49%	2
SURICATA ICMPv4 unknown code	1.49%	2
GPL SMTP RCPT TO overflow	0.75%	1
GPL SMTP vrfy root	0.75%	1
ET SCAN Potential SSH Scan	0.75%	1

Obrázek 6.2: Procento a počet detekovaných signatur v testu

Na následujících řádcích můžeme vidět aktivní prevenci Suricaty vůči útokům. Záznamy pocházejí z logu fast.log. Obrázek 6.3 pak ukazuje počet všech událostí během fáze testování.

```
04/11/2015-22:22:10.186024  [Drop] [**] [1:2012726:4] ET SCAN
OpenVAS User-Agent Inbound [**] [Classification: Attempted
Information Leak] [Priority: 2] {TCP} 2.0.0.2:33484 ->
1.0.0.2:80

04/11/2015-22:22:20.907357  [Drop] [**] [1:2101411:12] GPL SNMP
public access udp [**] [Classification: Attempted Information
Leak] [Priority: 2] {UDP} 2.0.0.2:38657 -> 1.0.0.2:161

04/11/2015-22:22:22.563123  [Drop] [**] [1:2101413:11] GPL SNMP
private access udp [**] [Classification: Attempted Information
Leak] [Priority: 2] {UDP} 2.0.0.2:51513 -> 1.0.0.2:161

04/11/2015-22:22:22.670676  [Drop] [**] [1:2011011:2] ET SNMP
Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden Read/Write
Community String ILMI [**] [Classification: Attempted
Administrator Privilege Gain] [Priority: 1] {UDP} 2.0.0.2:50036
-> 1.0.0.2:161

04/11/2015-22:22:23.033404  [Drop] [**] [1:2011013:2] ET SNMP
Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden Read/Write
Community String cable-docsis [**] [Classification: Attempted
```

```
Administrator Privilege Gain] [Priority: 1] {UDP} 2.0.0.2:45176
-> 1.0.0.2:161

04/11/2015-22:22:52.084340  [Drop] [**] [1:2101446:7] GPL SMTP
vrfy root [**] [Classification: Attempted Information Leak]
[Priority: 2] {TCP} 2.0.0.2:49482 -> 1.0.0.2:25

04/11/2015-22:23:16.606526  [Drop] [**] [1:2010935:2] ET POLICY
Suspicious inbound to MSSQL port 1433 [**] [Classification:
Potentially Bad Traffic] [Priority: 2] {TCP} 2.0.0.2:42925 ->
1.0.0.2:1433

04/11/2015-22:23:25.332548  [Drop] [**] [1:2200025:1] SURICATA
ICMPv4 unknown code [**] [Classification: (null)] [Priority: 3]
{ICMP} 2.0.0.2:8 -> 1.0.0.2:123

04/11/2015-22:23:30.666486  [Drop] [**] [1:2200075:1] SURICATA
UDPv4 invalid checksum [**] [Classification: (null)] [Priority:
3] {UDP} 2.0.0.2:53 -> 1.0.0.2:65533

04/11/2015-22:24:26.763168  [Drop] [**] [1:2100654:17] GPL SMTP
RCPT TO overflow [**] [Classification: Attempted Administrator
Privilege Gain] [Priority: 1] {TCP} 2.0.0.2:52800 -> 1.0.0.2:25
```



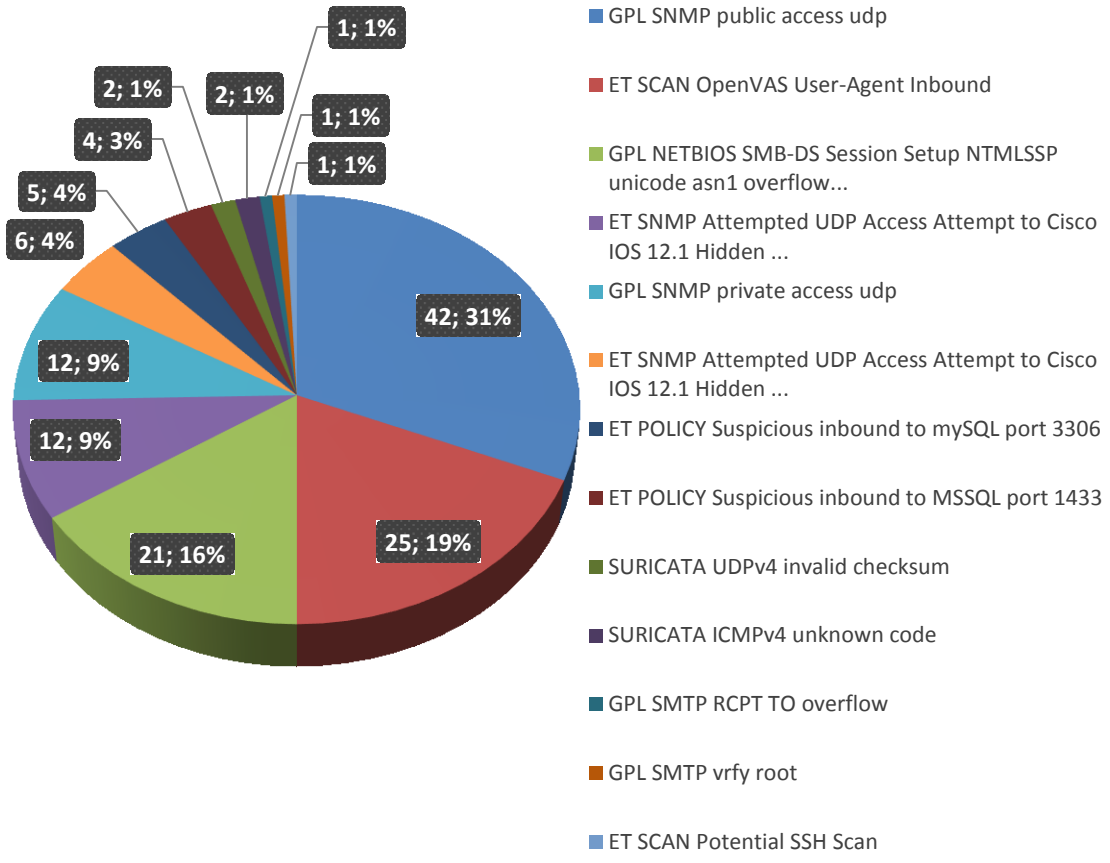
Obrázek 6.3: Graf ze Snorby znázornující počet událostí po testu v režimu IPS

6.5 Grafická analýza útoků v režimu IPS

6.5.1 Testování pomocí penetračního nástroje OpenVAS

V režimu IPS se počet detekovaných signatur rapidně zredukoval. Hlavní příčinou tohoto poklesu je princip testování z pohledu penetračního frameworku OpenVAS. Ten, jak jsem psal v předchozí kapitole, využívá ke svým testům další podpůrné služby, které předcházejí ostrému testování. Tím pádem nedojde na přímé testování a je aktivně útok zablokován. Komplexní přímé útoky využívající více typů útoků můžou být také blokovány už na začátku, a tudíž nedojde ke kompletnímu testování. I v praxi může útočník využívat komplexní skripty pro penetraci systému a blokování útoku z počátku může odvrátit rizika případného kompletního napadení.

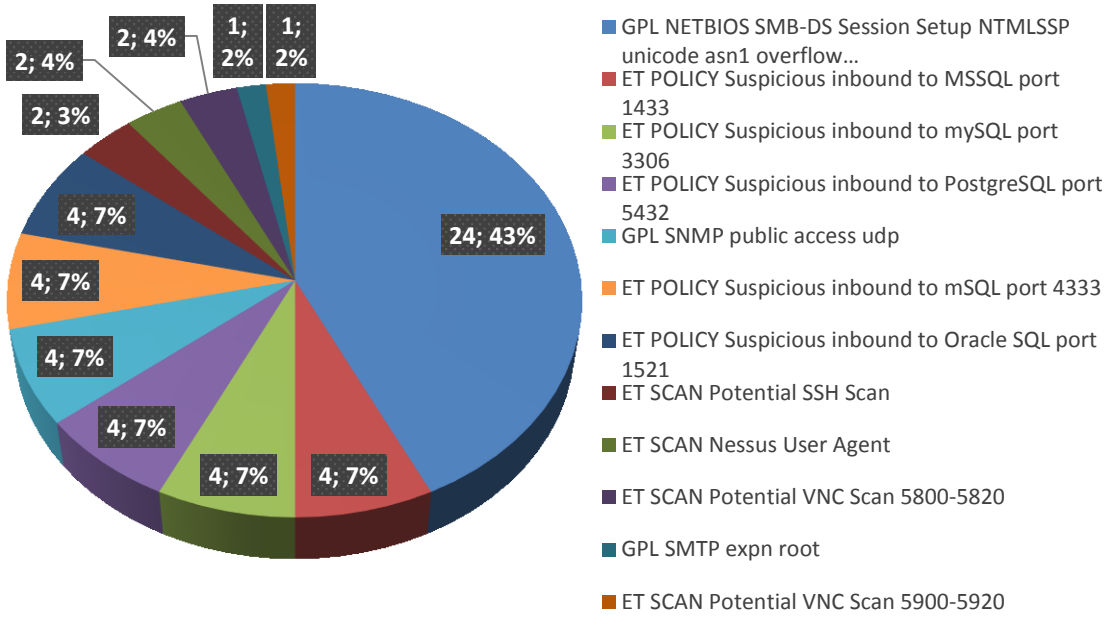
Graf 6.1 zobrazuje procentuální rozdělení signatur z testování v režimu IPS. Jak můžeme vidět počet signatur OpenVAS klesl z 76% na 19% oproti testování v IDS režimu. Popisky jednotlivých části grafu udávají absolutní hodnotu, což je počet signatur a druhá cifra udává její relativní hodnotu.



Graf 6.1: Procento detekovaných signatur v režimu IPS - OpenVAS

6.5.2 Testování pomocí penetračního nástroje Nessus

Oproti režimu IDS, kde byla nejčetnější signatura GPL SNMP public access udp, je v režimu IPS zastoupená jen sedmi procenty, Graf 6.2. Nejčetnější signatura byl určena pro testování Windows systému, opět podobná analogie jako u testování s OpenVAS, blokování některých testů způsobí, že jsou opakovány až do vypršení vnitřního intervalu.



Graf 6.2: Procento detekovaných signatur v IPS režimu - Nessus

6.6 Shrnutí údajů pro Nessus a OpenVAS

Tabulka 6.1 zobrazuje rozdělení všech signatur podle detekce penetračními nástroji. Hodnoty ve sloupcích IDS a IPS udávají počty událostí detekovaných v daném testu. Sloupec severity udává míru závažnosti dané signatury a poslední sloupec udává počet unikátně detekovaných signatur pro daný penetrační nástroj.

Značný úbytek a dokonce některé chybějící signatury v režimu IPS jsou důsledkem zahazování paketů, jak je vysvětleno v předchozí kapitole. Čeho si však můžeme všimnout je naopak navýšení počtu některých signatur v režimu IPS. To může být dáno snahou provést test, i když druhá strana neodpovídá, až do vypršení daného intervalu. Zahazování paketů typu DROP vypadá z pohledu útočníka jako nedostupnost cíle.

Penetration tool	Signature	Number of events		Severity	Number of signature
		IDS	IPS		
Nessus	ET POLICY Suspicious inbound to mSQL port 4333	3	4	2	8
	ET POLICY Suspicious inbound to Oracle SQL port 1521	3	4	2	
	ET POLICY Suspicious inbound to PostgreSQL port 5432	3	4	2	
	ET SCAN Nessus User Agent	2	2	2	
	ET SCAN Potential VNC Scan 5800-5820	2	2	2	
	ET SCAN Potential VNC Scan 5900-5920	1	1	2	
	GPL NETBIOS SMB-DS ADMIN$ unicode share access	1		3	
	GPL NETBIOS SMB-DS IPC$ unicode share access	5		3	
OpenVAS	GPL SMTP RCPT TO overflow	1	1	1	9
	GPL SNMP private access udp	12	12	2	
	GPL WEB_SERVER .htaccess access	42		2	
	GPL WEB_SERVER 403 Forbidden	3		2	
	SURICATA ICMPv4 unknown code	2	2	null	
	SURICATA UDPv4 invalid checksum	2	2	null	
	ET SNMP Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden ...	12	12	1	
	ET SNMP Attempted UDP Access Attempt to Cisco IOS 12.1 Hidden ...	6	6	1	
	ET SCAN OpenVAS User-Agent Inbound	364	25	2	
Nessus + OpenVAS	ET POLICY Suspicious inbound to MSSQL port 1433	3	4	2	7
	ET POLICY Suspicious inbound to mySQL port 3306	3	4	2	
	ET SCAN Potential SSH Scan	2	2	2	
	GPL NETBIOS SMB-DS Session Setup NTMLSSP unicode asn1 overflow…	10	24	3	
	GPL SMTP expn root	6	1	2	
	GPL SMTP vrfy root	5		2	
	GPL SNMP public access udp	28	4	2	
	CELKEM	521	116		24

Tabulka 6.1: rozdělení všech signatur podle odhalení penetračními nástroji

Závěr

Cílem této práce bylo komplexně otestovat a ověřit možnosti systému detekce vniknutí do počítačové sítě SURICATA. Testování probíhalo za pomocí různých virtualizačních a simulačních nástrojů, jako jsou VirtualBOX a GNS3. Pro testování byly použity 4 virtuální stroje, vytvořené v programu VirtualBOX s operačními systémy Linux a Windows. Linuxové distribuce reprezentovaly dva operační systémy UBUNTU 14.04 a jeden KALI OS 1.0.9. Jako zástupce pro Windows byl zvolen operační systém používaný pro servery, Windows Server 2008 R2 SP1 Datacenter. Virtuální stroje s OS Ubuntu a Windows byly využívány pro testování zranitelnosti a penetraci systému.

Systém Suricata byl otestován z několika pohledů. Jedno z hlavních hledisek bylo testování na efektivitu a výkonnost samotné detekce jak v režimu IDS nebo IPS. Pro tyto účely byly vytvořeny jednotlivé testy v testovacím frameworku OpenVAS a Nessus. Ze 14 kategorií byly vybrání jejich hlavní zástupci. Takto vytvořený testovací scénář byl spouštěn proti testovaným virtuálním strojům. Síťová topologie byla navrhnutá a používána k testování v programu GNS3.

Dalším pohledem je poukázání na možnosti vztahující se analýzy a exportu dat. Suricata nabízí export výstupů do formátu unified2 čitelnou pro databázi, například MySQL. Data z tohoto je ale potřeba importovat do databáze. Na řadu tak přichází program Barnyard2, který působí jako prostřední mezi Suricatou a databází. Tato databáze je pak interní databází různých grafických monitorovacích rozšíření, jako byl v našem případě testovaný Snorby. Z této databáze interpretuje vložená data Barnyardem graficky.

Testování na efektivitu a výkonnost přineslo velmi pozitivní výsledky z hlediska detekce útoků. Můžeme říct, že pravidla vytvářena společností Emerging Threats měla téměř 100% účinnost. To však nemůžeme říct o pravidlech VRT pro konkurenční Snort. Kompatibilita s těmito pravidly není vůbec dobrá. Už při importu pravidel Suricata oznamoval několik chyb. Pouze necelých 50% pravidel se korektně importovalo. Při testu však žádné anomálie nedetekovala. Proto tyto pravidla nedoporučuji pro testování, ale zato doporučuji oficiální pravidla společnosti Emerging Threats. Výsledky testování byly srovnány a zhodnoceny také graficky.

Použitá literatura

[1] ŘEZÁČ, F. M. VOZŇÁK a J. ROZHON. *Bezpečnost v komunikacích*. Ostrava: 2013.

[2] GIBSON, D. *CompTIA Security+: Get Certified Get Ahead: SY0-301 Study Guide*. 2011. ISBN 978-1463762360.

[3] GRYGÁREK, P. *Bezpečnost počítačových sítí*. Ostrava. Dostupné také z: http://www.cs.vsb.cz/grygarek/SPS/lect/bezpecnost-ucitele.pdf

[4] PETER, T. a D. MICHALEC. VSB. In: *IPSec* [online]. [cit. 2014-08-22]. Dostupné z: http://www.cs.vsb.cz/grygarek/TPS-0304/projekty0304/ipsec/ipsec.html

[5] ZELINKA, A. *Paketové filtry*. Brno: 2002. Masarykova Univerzita, Katedra infromatiky [cit. 2014-Srpen-30]. Dostupné z: http://www.fi.muni.cz/~kas/p090/referaty/2002-podzim/skupina10/firewall.html

[6] CASWELL, B. *Snort IDS and IPS Toolkit (Jay Beale's Open Source Security)*. ISBN:978-1597490993.

[7] *Computer Application and System modeling*. Taiyuan: Institute of Electrical and Electronics Engineers, Inc. [cit. 2014-10-05]. ISBN 978-1-4244-7236-9. Dostupné z: http://cs.wikipedia.org/w/index.php?title=ARP_spoofing

[8] LANG, J.P. openinfosecfoundation.org. In: *Suricata.yaml* [online]. 2015. Dostupné také z: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml

[9] OPENVAS. openvas.org. In: *About OpenVAS Software* [online]. Dostupné také z: http://www.openvas.org/software.html#standards_and_interoperability

[10] www.zive.cz. *Bezpečnost - Živě.cz* [online]. 2015. Dostupné také z: http://www.zive.cz/bezpecnost/sc-120/default.aspx?tags=1

[11] www.soom.cz. *SOOM.cz - Aktuality* [online]. 2015. Dostupné také z: http://www.soom.cz/aktuality

[12] AMOROSO, E. *Cyber Attacks: Protecting National Infrastructure*. 2010. ISBN: 978-0123849175.

[13] BEJTLICH, R. *The Tao of Network Security Monitoring: Beyond Intrusion Detection*. 2004. ISBN: 978-0321246776.

[14] ERICKSON, J. *Hacking: The Art of Exploitation, 2nd Edition*. No Starch Press, 2008. ISBN: 978-1593271442.

[15] ENGEBRETSON, P. *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. 2011. ISBN: 978-1597496551.

[16] COX, K. J. a G. CHRISTOPHER. *Managing Security with Snort and IDS Tools*. ISBN 978-0596006616.

[17] GREGG, M. a B. HAINES. *CASP: CompTIA Advanced Security Practitioner Study Guide Authorized Courseware: Exam CAS-001*. 2012. ISBN: 978-1118083192.

[18] MACHNÍK, P. *Širokopásmové sítě pro integrovanou výuku VUT a VŠB-TUO*.. Ostrava: Vysoká škola báňská-Technická univerzita Ostrava, 2014. 978-80-248-3630-0.

[19] ALLEN, L. *Advanced Penetration Testing for Highly-Secured Environments: The Ultimate Security Guide*. 2012. ISBN: 978-1849517744.

[20] HADNAGY, C. *Social Engineering: The Art of Human Hacking*. 2010. ISBN: 978-0470639535.

[21] Every Day is Zero Day. *Installing Snorby on Ubuntu 12.04* [online]. Dostupné také z: http://blog.beor.co.za/2013/01/installing-snorby-on-ubuntu-1204.html

[22] GitHub. *Snorby Worker Troubleshooting* [online]. Dostupné také z: https://github.com/Snorby/snorby/wiki/Snorby-Worker-Troubleshooting

[23] Askubuntu. *Problem installing snort barnyard2* [online]. Dostupné také z: http://askubuntu.com/questions/189722/problem-installing-snort-barnyard2

[24] Suricata. *Suricata,Snorby and Barnyard2 set up guide* [online]. Dostupné také z: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/suricata_snorby_and_barnyard2_set_up_guide

[25] FRITZ, S. Stephen Fritz on Systems Engineering. *Linux with Suricata, Barnyard2 and Snorby* [online]. Dostupné také z: http://stephenfritz.blogspot.cz/2014/05/linux-with-suricata-barnyard2-and-snorby.html

[26] Suricata Wiki. *Rule Management with Oinkmaster* [online]. Dostupné také z: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Rule_Management_with_Oinkmaster

[27] Everything Should Be Virtual. *Suricata IDS/IPS VMXNET3* [online]. Dostupné také z: http://everythingshouldbevirtual.com/suricata-idsips-vmxnet3

[28] Suricata Wiki. *Setting up IPS/inline for Linux* [online]. Dostupné také z: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Setting_up_IPSinline_for_Linux

Seznam příloh

Součástí diplomové práce je CD/DVD. Ve složkách Suricata se nachází všechny záznamy z penetračního testování. Součástí složek IDS a IPS je také report z webového monitoringu Snorby a konfigurační skript pro IPS režim.

Adresářová struktura přiloženého CD/DVD:

```
+---Nessus
|   +---IDS
|   |   \---suricata
|   +---IPS
|       \---suricata
+---OpenVAS
|   +---IDS
|   |   \---suricata
|   +---IPS
|   |   \---suricata
```